Engineering memo

Content

1 Mbed projects....5
1.1 Mikroe TFT-Proto....5
1.1.1 TFT specifications....5
1.1.2 Setting Mbed and MikroeTFT-Proto....7
1.1.3 PWM controlled brightness....11
1.1.4 Building keyboard....13
1.1.5 Loading image to Mikroe TFT-Proto....31
1.2 ER-TFT070, 7'' TFT-touch....33
1.2.1 MBED – Textlists with automated touch action slots....41
1.3 XBEE....45
1.4 RFID....46
1.4.1 RFID Click....46
1.4.2 RDM6300....49
2 Teensy++....54
2.1 Required equipment/tools....54
2.2 Setting up project....54
2.3 Interrupt Timer....55
2.4 UART communication....56
2.5 Compare two strings....59
2.6 Convert string to integer....60
2.7 Connecting Teensy to Xbee....61
2.8 Hope RFM23BP (Still in progress)....62
3 Atmel AVR....67
3.1 Timers....67
3.2 Configuring timers....67
3.2.1 One second timer 0 example with 8 bit timer....68
3.2.2 One second timer 1 example with 16 bit timer....69
3.2.3 1mS Overflow Interrupt timer with timer 1....70
4 Arduino....71
4.1 Choosing development board....72
4.2 Blink led....73
4.3 Async led blink with counter....73
4.4 USB Serial hello world....74
4.5 USB Serial echo....74
4.6 Timing systems....75

4.6.1 Timing systems with constant time display....80
4.6.2 Split timer....82
4.7 Distance meter used for measuring jump heights....83
4.8 Radio Module: RF4432F27 500mW 868MHz....92
4.9 GSM/GPRS with NEOWAY M590....93
4.9.1 Connect pc serial terminal with NEOWAY....93
4.9.2 Send SMS....96
5 Circuit Design....100
5.1 Basics....100
5.1.1 How to read schematics....100
5.2 KiCad....102
5.2.1 Creating custom symbol for component....102
5.3 Power Supply....103
5.4 Swithcing power supply: Buck converter simple....103
5.5 Swithcing power supply....107
5.6 1-Cell chargher design for Li-Ion / Li-Polymer Charge....107
6 Circuit Protection....110
6.1 Over-voltage protection....111
6.2 Over-current protection....111
6.3 Over-discharge protection....113
6.4 Lithium battery protection circuit....116
7 Jetson....118
7.1 Flash....118
7.2 Set SPI....118
7.3 UART communication HM-10 BLE....120
8 Software programming....122
8.1 Django rest framework on EC2 instance....122
8.1.1 Django basics....122
8.1.1.1 Importing modules with same name....122
8.1.1.2 Script to run and combile your Django....122
8.1.2 Starting and accessing EC2 instance....123
8.1.3 Installing Django REST framework....124
8.1.4 Create a project....124
8.1.5 Creating first application....124
8.1.6 Adding token authentication....128
8.1.7 Creating user and generating token....130

8.1.8 Creating user and generating token through end point....131
8.2 Install CUDA & CUDNN....133
8.3 OpenCV 3.4.0 – Open source computer vision library....134
8.3.1 Install OpenCV on ubuntu....134
8.4 Object detection with Darknet YOLO....136
8.4.1 Install darknet....136
8.4.2 Running YOLO and detecting objects....137
8.4.3 Training YOLO to recognize custom objects....137
9 Investment & Financing....140
10 Examples #TODO move these to seperate files....142
10.1 Keyboard example code....142

1 Mbed projects

1.1 Mikroe TFT-Proto

1.1.1 TFT specifications

Color Display:	MI0283QT-9A.
Screen:	320x240 pixel.
Display controller:	ILI9341.
Color:	262K RGB.
Data:	8 Data lines and 5 control lines.
Touch:	4-wire resistive touch panel.
Operating voltage:	3.3V power supply. 5V recommended for TFT back light.

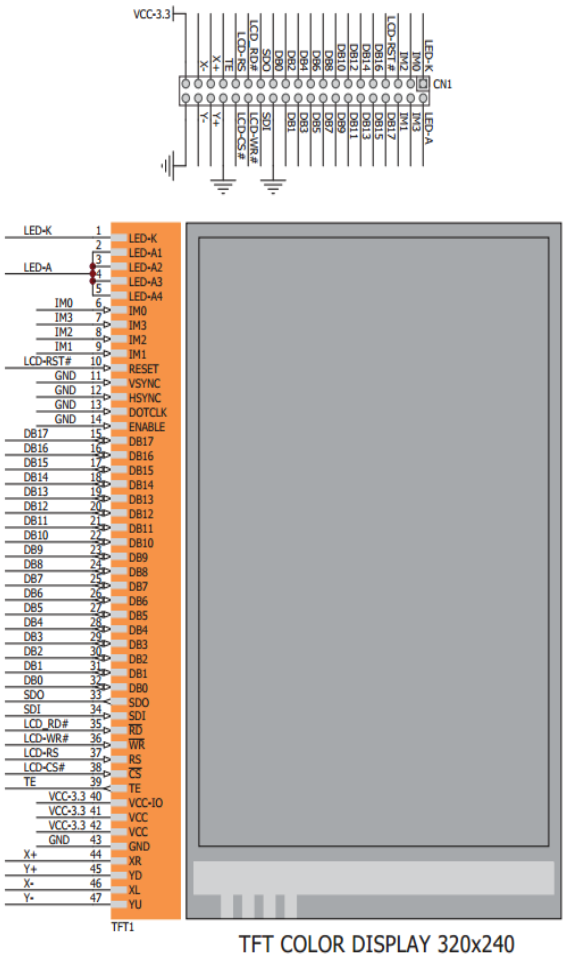


Figure 2: TFT PROTO schematic

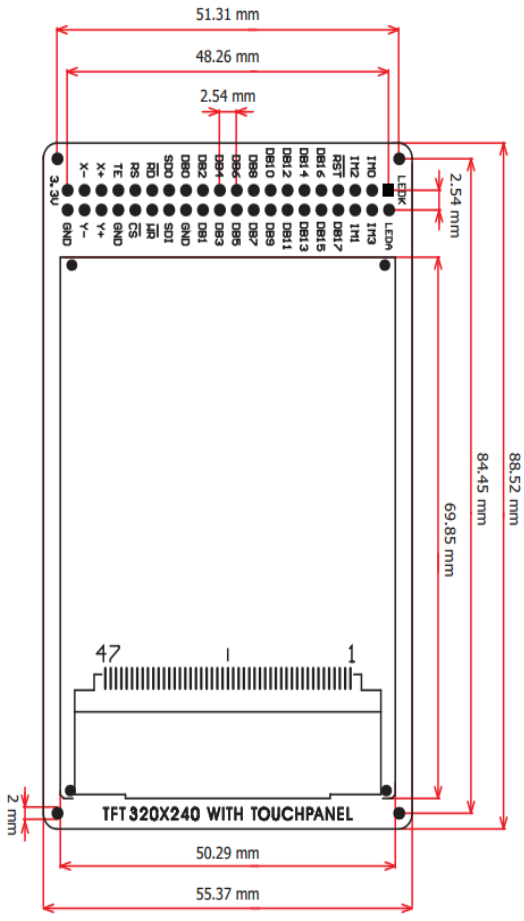


Figure 3: TFT PROTO dimensions

Kuva 1: TFT Proto schematic & dimensions, page 3.[1]

Mbed specifications:

Hint!. Mbed has 32kbytes ram so when building bigger scale programs it’s recommended to use as few arrays as possible and save ram using type char. Char uses 1 byte per data while using int requires 2 bytes.

On a 32 bit system it is as:

char : 1 byte

short : 2 bytes

int : 2 bytes

long : 4 bytes

float : 4 bytes

double : 8 bytes

On a 64 bit system:

char : 2 byte

short : 4 bytes

int : 4 bytes

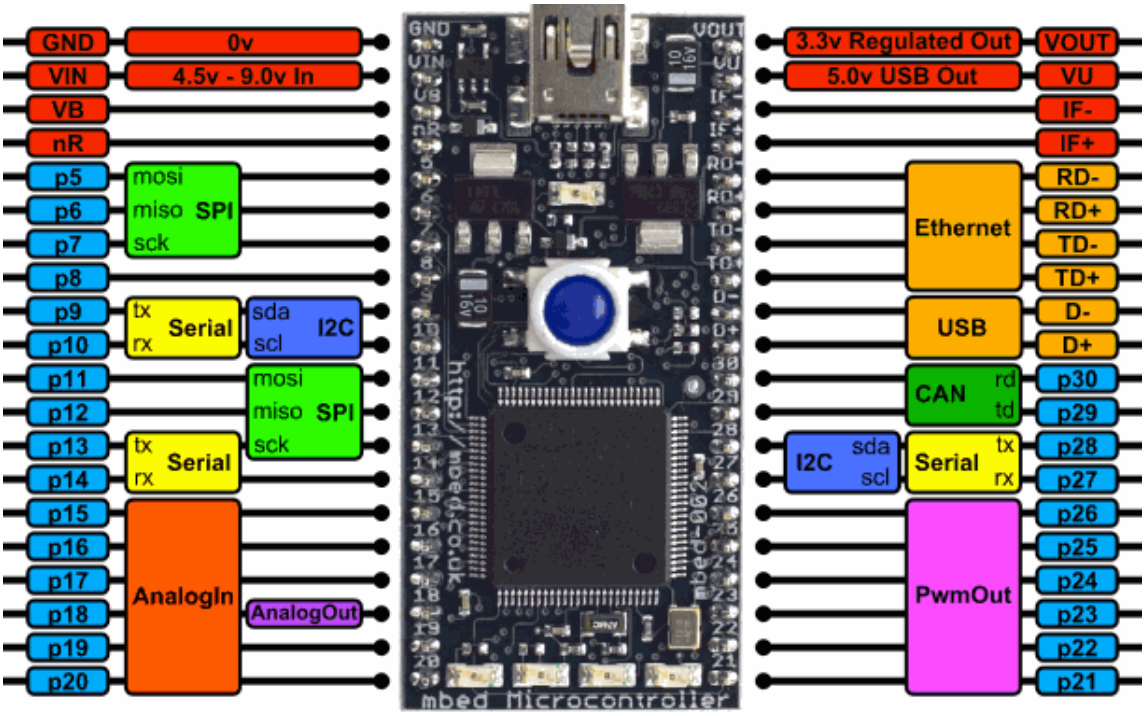
long : 8 bytes

float : 8 bytes

double : 16 bytes

NXP LPC1768

Memory Flash 512Kb Ram 16/32Kb



Kuva 2: Mbed GPIO pin table.[15]

1.1.2 Setting Mbed and MikroeTFT-Proto

Mbed to display p20 (x+), p19 (x-) , p18 (y+) , p17 (y-) , p11 (mosi) , p12 (miso) , p13 (sck) , p14 (cs) , p15 (reset) , p21 (dc)

MBED	Display
+ 3,3V	3,3V
GND	GND
mosi	SDI
miso	SDO
sck	RS
cs	CS
reset	RST
dc	/WR
GND	IM0
+3,3V	IM1 IM2 IM3
GND	DB0 - DB17
GND	RD

Backlight: Plug LED A to +5. Put a 2.5R resistor or 2x5R resistor in parallel between LED K and GND.

Get started by creating new program. Click “New” icon at left top corner of MBED compiler. Select New Program. Select “Empty Program” from Template: list. By right clicking your new program folder at Program Workspace you get a list open. From this list select New File and name it as main.cpp to make program file.

Import Michael Ammanns http://mbed.org/users/mazgch/code/SeeedStudioTFTv2/ library. Import this library to your program folder. Select your program from “Target Path”. Plug x+, x- and y+, y- to analog in pins. SPI_TFT_ILI9341 library included.

By selecting SPI_TFT_ILI9341 from imported SeeedStudioTFTv2 library, you can check library version. If the SPI_TFT_ILI9341 library is out of date, you can see under Program Detail “The documentation is out of date”. In that case download latest library from http://mbed.org/users/dreschpe/code/SPI_TFT_ILI9341/ otherwise not necessary.

Import fonts from http://mbed.org/users/dreschpe/code/TFT_fonts/ to your program folder. Now you should have in Program folder -> libraries SeedStudioTFTv2, TFT_fonts, and your main.cpp (Names can vary).

After importing touch and display files, open SeedStudioTFTv2 library -> SeeedStudioTFTv2.h -> replace

```
SeeedStudioTFTv2(PinName xp, PinName xm, PinName yp, PinName ym,
                 PinName mosi, PinName miso, PinName sclk,
                 PinName csTft, PinName dcTft, PinName blTft,
                 PinName csSd);
```

with

```
SeeedStudioTFTv2(PinName xp, PinName xm, PinName yp, PinName ym,
                 PinName mosi, PinName miso, PinName sclk,
                 PinName csTft, PinName dcTft, PinName blTft);
```

SeedStudio library was made for different screen which has integrated SD-card so we need to remove this to adapt it for Mikroe TFT-Proto.

Scroll down SeeedStudioTFTv2.h file until you find protected:. Under protected, cut Touch getTouch(point& p); typedef enum { YES, MAYBE, NO } TOUCH;

Move these under public:. Content under public can be called and modified in other .cpp files.

```
public:
TOUCH getTouch(point& p);
typedef enum { YES, MAYBE, NO } TOUCH;
```

Now open SeeedStudioTFTv2.cpp -> remove csSD from

```
SeeedStudioTFTv2::SeeedStudioTFTv2(PinName xp, PinName xm, PinName yp,
PinName ym,
                                   PinName mosi, PinName miso, PinName sclk,
                                   PinName csTft, PinName dcTft, PinName blTft,
                                   PinName csSd):
```

Now it should look like this

```
SeeedStudioTFTv2::SeeedStudioTFTv2(PinName xp, PinName xm, PinName yp,
PinName ym,
                                   PinName mosi, PinName miso, PinName sclk,
                                   PinName csTft, PinName dcTft, PinName blTft):
```

Under

#ifndef USE_SDCARD set the code into comments as follows

```
    // sd card
```

```
    //DigitalOut cs(csSd);
    //cs = 1;
```

Now open up main.cpp

Start by organizing structure.

Libraries

Defination

Variables

Possible sub-programs

main program

When working on code keep things in order and structured. This helps in debugging.

Example code:

```
#include “mbed.h”
#include "SeeedStudioTFTv2.h"
// fonts
#include “Arial12x12.h”
#include “Arial24x23.h”
#include “Arial28x28.h”
#include “font_big.h”
// Serial connection between PC and MBED
Serial pc(USBTX, USBRX);
// x+,x-,y+,y-,mosi, miso, sclk, cs, dc, reset
SeeedStudioTFTv2 tft(p19,p18,p17,p16,p11, p12, p13, p14, p22, p20);
/*For driving display through SPI p4-p7 or p11 - p13 can be used on mbed.
“cs, reset, dc” pins can be freely set to any mbed pinouts.
x+, x-, y+, y- has to be set to mbed analog inputs.*/
point p;       // creates pointer p.x,p.y

// sub-program
```

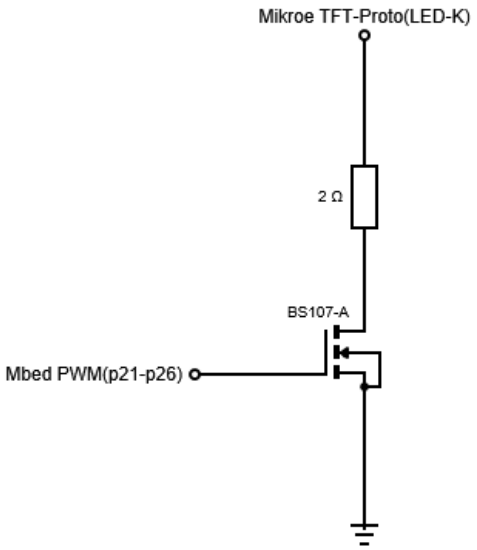
```
// this program will wait until user touches screen
// analog data of touch panel is converted trough ADC to digital value
// getTouch will store that value
//getPixel will then convert ADC value to 320*240 pixel coordinates
// this touch program will be used in every code at this example

void touch()
 {
   while(1)
  {
     p.x=0;p.y=0;
     if (tft.getTouch(p)==tft.YES) // read analog pos.
     {
        tft.getTouch(p);
        tft.getPixel(p);            // convert to pixel pos
        wait_ms(150);     // give some time for the touch
        break;
       }
   }
}

void main(void)
{
                //TFT initialize
                tft.claim(stdout);      // send stdout to the TFT display
                tft.set_orientation(1);    // set screen orientation
                tft.background(Black);    // set background to black
                tft.foreground(White);    // set chars to white
                tft.cls();               // clear the screen
                tft.set_font((unsigned char*) Neu42x35);          // set used txt font
                tft.locate(70,100);        // Set pixel location at screen
                tft.printf("The Duuude ");              // Print txt to designated pixe location
```

```
        tft.set_font((unsigned char*) Arial12x12);
        tft.locate(209,120);
        tft.printf("So stoked");
        wait_ms(300);              // wait for 300ms
        tft.set_font((unsigned char*) Arial24x23);
        tft.locate(50,155);
        tft.printf("Welcome to coding mates");
        wait(2); // wait 2 seconds
}
```

1.1.3 PWM controlled brightness

Mikroe TFT-Proto brightness can be controlled by manually changing resistor value or through mbed using pwm output. When we drive pwm signal to mosfet, we can easily change the brightness by changing duty cycle of pulse. This will create illusion for eyes. What happens is pwm will be turning TFT backlight on and off, if this is done fast enough human eyes won't be able to notice screen blinking. Instead this will be seen as dimmer- or brighter screen even thou every high pulse gives full brightness for the screen. Basicly by changing the duty cycle of period we determine the brightness of screen.

The BS107-A mosfet has Rds 4.5 to 6.5 Ohm internal resistance. Rds stands for resistance drain to source. Rds varies depending on mosfet, you can check it from datasheet.



Code->>

Define:

```
PwmOut Brightness(p21);              // pwm output
```

Rest comes inside Main()

```
Brightness = 1;            // Set brightness control on. Now brightness is at highest
Brightness.period(0.005);              // set period of pulse
```

Slider for controlling pwm value.

Defination:

```
char cl = 5;                // Value for slider at maximum brightness
```

Next function will be putted under void main()

Create image for brightness slider

```
tft.fillrect( 151, 136, 266, 146, Black);
// this determines where is the slider bar going to be located at the brightness bar
// ( ( x-axis slider start + ( slider start point * brightness level ) ), ( y-axis bar begins from 136 ) , ( ( X-axis slider end + ( Slider width * brightness lvl ) ) ), Color );
tft.fillrect( 127 + ( 24 * cl), 136, 146 + ( 24 * cl), 146, Red);
```

Function:

```
touch();
float tempCS = p.x; // copy x-axis value
//  ( ( ( ( present x-axis value – brightness bar length ) / (brightness bar length – slider bar width) ) * brightness level(s) ) + 1( to keep slider bar inside brightness bar )
float CSliderf = ( ( ( ( tempCS – 150 ) / 117 ) * 5) + 1 );
cl = int( CSliderf );        // convert Csliderf value to int
// pc.printf(" CSliderf %f: cl:%d", CSliderf, cl); // Debugging
Brightness = CSliderf*0.2;              // set brightness x*(1/5)
```

1.1.4 Building keyboard

Building keyboard to mbed/mikroe_tft-proto.

!!point(x,y) x= pixel on x axis, y pixel on y axis.

!!Full code example at Examples section.(Can be copied straight)

Step 1.

Lets use the keyboard sketch on attachment 2 or draw your own(better option). The keyboard is divided to two different areas textbox and keyboard. Keep textbox clean, this is where you print all written characters. For keyboard make square slot for each character you want to print. When all previous is done, calculate pixel location for each slot corners. After calculating and locating every character, we move on to actual programming.

Step 2.

Open mbed website -> compiler → new project. Start by defining variables and splitting screen into keyboard area and character display area. Mikroe-TFT's resolution 320x240-> 320 width 240 height.

Define touchscreen pins. Here we call it tft.

```
touch_tft tft(p19,p18,p17,p16,p11, p12, p13, p14, p21, p22,"TFT");  //
x+,x-,y+,y-,mosi, miso, sclk, cs, reset, dc
```

Build the keyboard in small steps, check the screen often for correct positioning. Lets use char type defination for minimal memory usage. (char = 1byte int = 2byte)

```
// Keyboard Variables
char buffer[124];          // Buffer for storing pushed keys
char key;                  // Used for deleting and adding characters to buffer
char chaty;                // Used for defining print location for characters in y axis
char key_back[124];        // Used for defining print location and for backspace in x axis
char key_i;                // Array location
char t, t2, t3, t4, t5, t6; // Keyboard functionality variables
char key_skip;

// Keyboard Frame
tft.background(Black);  // Background colour
tft.foreground(white);  // Character colour
```

Choose text box and keyboard area by defining x(Vertical) and y(Horizontal).

```
tft.fillrect(0,0,320,120, LightGrey);  // Variety of colour can be used.
```

This will now be our text box. Next step keyboard background.

```
tft.fillrect(0,120,320,120, Black);     // Keyboard background
```

Start drawing lines to the keyboard area. Create slots for letters and numbers.

```
// Keyboard frame
tft.rect(0,120,320,160, Red); // draw line 1 & 2
tft.rect(0,190,320,220, Red); // draw line 3 & 4
```

Here I'm following the keyboard papersheet. When following coordinates at the paper it is faster to get all slots to right position. Remember different fonts and Capital/small letters affect slot size. Capital letters requires most of the space so we use capital letters to determine slot size. Example of keyboard coordinate paper sheet at attachment 2.

Drawing slots for first row (Now y axis value has been changed to 150-180)

```
tft.line(32, 120, 32, 150, Red); // tt3.line(x(start), y(start) ,x(end) ,y(end) ,color)
tft.line(64, 120, 64, 150, Red);
tft.line(96, 120, 96, 150, Red);
tft.line(128, 120, 128, 150, Red);
tft.line(160, 120, 160, 150, Red);
tft.line(192, 120, 192, 150, Red);
tft.line(224, 120, 224, 150, Red);
tft.line(256, 120, 256, 150, Red);
tft.line(288, 120, 288, 150, Red);
```

Drawing slots for second row (Now y axis value has been changed to 150-180).

```
tft.line(16, 150, 16, 180, Red);
tft.line(48, 150, 48, 180, Red);
tft.line(80, 150, 80, 180, Red);
tft.line(112, 150, 112, 180, Red);
tft.line(144, 150, 144, 180, Red);
tft.line(176, 150, 176, 180, Red);
tft.line(208, 150, 208, 180, Red);
tft.line(240, 150, 240, 180, Red);
```

```
tft.line(272, 150, 272, 180, Red);
tft.line(304, 150, 304, 180, Red);
```

Drawing slots for third row (Now y axis value has been changed to 180-210).

```
tft.line(43, 180, 43, 210, Red);
tft.line(75, 180, 75, 210, Red);
tft.line(107, 180, 107, 210, Red);
tft.line(139, 180, 139, 210, Red);
tft.line(171, 180, 171, 210, Red);
tft.line(203, 180, 203, 210, Red);
tft.line(235, 180, 235, 210, Red);
tft.line(267, 180, 267, 210, Red);
```

Drawing slots for fourth row (Now y axis value has been changed to 210-240).

```
tft.line(43, 210, 43, 240, Red);
tft.line(224, 210, 224, 240, Red);
tft.line(267, 210, 267, 240, Red);
```

Now we move on printing characters to the character slots. But before that we want to think how many different character sets we need. In this case we are using 3. Capital, lower-case, and numbers. We make a switch case with variable “set”. “set” presents the character set we want to show.

```
switch( set )
    {
    // Lowercase
    case 1:  // set equals 1 if Lower character are chosen
```

Printing characters of first row

```
            tft.locate(12, 130); tft.putc('q');          //tft.locate(x,y). When printing characters use ‘char‘
            tft.locate(45, 130); tft.putc('w');          //tft.putc(‘printed data’).
            tft.locate(77, 130); tft.putc('e');
            tft.locate(109, 130); tft.putc('r');
```

```
tft.locate(141, 130); tft.putc('t');
tft.locate(174, 130); tft.putc('y');
tft.locate(205, 130); tft.putc('u');
tft.locate(237, 130); tft.putc('i');
tft.locate(269, 130); tft.putc('o');
tft.locate(301, 130); tft.putc('p');
```

Printing characters of second row

```
tft.locate(28, 160);  tft.putc('a');
tft.locate(60, 160);  tft.putc('s');
tft.locate(92, 160);  tft.putc('d');
tft.locate(124, 160); tft.putc('f');
tft.locate(156, 160); tft.putc('g');
tft.locate(188, 160); tft.putc('h');
tft.locate(220, 160); tft.putc('j');
tft.locate(252, 160); tft.putc('k');
tft.locate(284, 160); tft.putc('l');
```

Printing characters of third row

```
tft.locate(10, 190);  tft.printf("A/a");      // Here we get to select Upper or Lower
    case tft.printf(“string”)
tft.locate(55, 190);  tft.putc('z');
tft.locate(87, 190);  tft.putc('x');
tft.locate(118, 190); tft.putc('c');
tft.locate(151, 190); tft.putc('v');
tft.locate(183, 190); tft.putc('b');
tft.locate(215, 190); tft.putc('n');
tft.locate(247, 190); tft.putc('m');
tft.set_font( ( unsigned char* ) Arial24x23);  // Set font
tft.locate(273, 186); tft.printf("<~");     // Represents backspace
```

More fonts can be gained from mbed website.

Printing characters of third row

```
tft.set_font( ( unsigned char* ) Arial12x12);
tft.locate(6, 220);   tt3printf("123");
tft.set_font( ( unsigned char* ) Arial24x23);
tft.locate(80, 215);  tft.printf("SPACE");
tft.locate(238, 215); tft.putc('.');
tft.locate(274, 215); tft.printf("<]");        // Represents enter
tft.locate(263, 50); tft.printf("OK");         // Pushing OK closes keyboard window
                                                and stores written data // to buffer
break;
```

In case 2 we are dealing with Capital letters. Now these require more space from screen to be drawn so the character slots are adjusted by Capital letters.

When case 2 is called, means user pushed A/a button.

```
case 2:

tft.locate(12, 130);  tft.putc('Q');
tft.locate(45, 130);  tft.putc('W');
tft.locate(77, 130);  tft.putc('E');
tft.locate(109, 130); tft.putc('R');
tft.locate(141, 130); tft.putc('T');
tft.locate(174, 130); tft.putc('Y');
tft.locate(205, 130); tft.putc('U');
tft.locate(237, 130); tft.putc('I');
tft.locate(269, 130); tft.putc('O');
tft.locate(301, 130); tft.putc('P');

tft.locate(28, 160);  tft.putc('A');
tft.locate(60, 160);  tft.putc('S');
tft.locate(92, 160);  tft.putc('D');
tft.locate(124, 160); tft.putc('F');
tft.locate(156, 160); tft.putc('G');
tft.locate(188, 160); tft.putc('H');
```

```
        tft.locate(220, 160); tft.putc('J');
        tft.locate(252, 160); tft.putc('K');
        tft.locate(284, 160); tft.putc('L');

        tft.locate(10, 190);  tft.printf("A/a");
        tft.locate(55, 190);  tft.putc('Z');
        tft.locate(87, 190);  tft.putc('X');
        tft.locate(118, 190); tft.putc('C');
        tft.locate(151, 190); tft.putc('V');
        tft.locate(183, 190); tft.putc('B');
        tft.locate(215, 190); tft.putc('N');
        tft.locate(247, 190); tft.putc('M');
        tft.set_font( ( unsigned char* ) Arial24x23);
        tft.locate(273, 186); tft.printf("<~");

        tft.set_font((unsigned char*) Arial12x12);
        tft.locate(6,220);   tt3.printf("123");
        tft.set_font((unsigned char*) Arial24x23);
        tft.locate(80,215);  tt3.printf("SPACE");
        tft.locate(238,215); tt3.putc('.');
        tft.locate(274,215); tt3.printf("<]");
        tft.locate(263, 50); tft.printf("OK");

break;        // End switch case and move on next function
```

In case 3 we deal with numbers. Capital letters should take more space than numbers, therefore adjusting character slots again is unnecessary.

```
 case 3:

        tft.set_font( ( unsigned char* ) Arial12x12);
```

Printing characters of first row.

```
tft.locate(12, 130);  tft.putc('1');
tft.locate(45, 130);  tft.putc('2');
tft.locate(77, 130);  tft.putc('3');
tft.locate(109, 130); tft.putc('4');
tft.locate(141, 130); tft.putc('5');
tft.locate(174, 130); tft.putc('6');
tft.locate(205, 130); tft.putc('7');
tft.locate(237, 130); tft.putc('8');
tft.locate(269, 130); tft.putc('9');
tft.locate(301, 130); tft.putc('0');
```

Printing characters of second row.

```
tft.locate(28, 160);  tft.putc('!');
tft.locate(60, 160);  tft.putc('#');
tft.locate(92, 160);  tft.putc('%');
tft.locate(124, 160); tft.putc('&');
tft.locate(156, 160); tft.putc('/');
tft.locate(188, 160); tft.putc('(');
tft.locate(220, 160); tft.putc(')');
tft.locate(252, 160); tft.putc('=');
tft.locate(284, 160); tft.putc('?');
```

Printing characters of third row.

```
tft.locate(10, 190);  tft.printf("A/a");
tft.locate(55, 190);  tft.putc('<');
tft.locate(87, 190);  tft.putc('>');
tft.locate(118, 190); tft.putc(',');
tft.locate(151, 190); tft.putc(';');
tft.locate(183, 190); tft.putc(':');
tft.locate(215, 190); tft.putc('-');
tft.locate(247, 190); tft.putc('_');
tft.set_font( ( unsigned char* ) Arial24x23);
```

```
    tft.locate(273, 186); tft.printf("<~");


    tft.set_font( ( unsigned char* ) Arial12x12);
    tft.locate(6, 220);   tft.printf("123");
    tft.set_font( ( unsigned char* ) Arial24x23);
    tft.locate(80, 215);  tft.printf("SPACE");
    tft.locate(238, 215); tft.putc('.');
    tft.locate(274, 215); tft.printf("<]");
    tft.locate(263, 50); tft.printf("OK");
    tft.background( LightGrey );
break;
 };
```

Now we just finished making the keyboard interface. Next we build touch input which stores written data to microprocessor.

Now we look from the flow chart what we need. We need one input for selecting character and two outputs one to screen, one to buffer. In this case our keyboard is touch screen, so we need pixel positions as input. Mikroe TFT uses 4 wire resistive system for determining touch location. This system gives data out as analog signal to our mbed analog inputs. Signal is then converted to pixel location on screen by Peters library.

Lets begin. Start by adding while loop. Inside the while loop we add request input(touch();) and all characters. This can be done with switch case or using if sentences. Lets use if sentences. Touch() function at page 8.

```
while(1)
   {
      touch();              // Request input

      // Keyboard first row
      switch(set)  //  Character set
      {
        tft.background( LightGrey ); // Change background color of characters
        tft.foreground(Black); // change caharcter color to black
```

```
        if( ( p.x >=0 && p.x <= 31) && ( p.y >= 120 && p.y <= 149 ) )   {key = 'Q';}; // Q

        if( ( p.x >= 32 && p.x <= 63 ) && ( p.y >= 120 && p.y <= 149 ) )  {key = 'W';};  // W

        if( ( p.x >= 64 && p.x <= 97 ) && ( p.y >= 120 && p.y <= 149 ) )  {key = 'E';};  // E

        if( ( p.x >= 98 && p.x <= 127 ) && ( p.y >= 120 && p.y <= 149 ) ) {key = 'R';};  // R

        if( ( p.x >= 128 && p.x <= 159 ) && ( p.y >= 120 && p.y <= 149 ) ) {key = 'T';};  // T

        if( ( p.x >= 160 && p.x <= 191 ) && ( p.y >= 120 && p.y <= 149 ) ) {key = 'Y';};  // Y

        if( ( p.x >= 192 && p.x <= 223 ) && ( p.y >= 120 && p.y <= 149 ) ) {key = 'U';};  // U

        if( ( p.x >= 224 && p.x <= 255 ) && ( p.y >= 120 && p.y <= 149 ) ) {key = 'I';};  // I

        if( ( p.x >= 256 && p.x <= 287 ) && ( p.y >= 120 && p.y <= 149 ) ) {key = 'O';};  // O

        if( ( p.x >= 288 && p.x <= 320 ) && ( p.y >= 120 && p.y <= 149 ) ) {key = 'P';};  // P


        // Keyboard second row

        if ( ( p.x >= 16 && p.x <= 47 ) && ( p.y >= 151 && p.y <= 179 ) )  { key = 'A';};  // A

        if ( ( p.x >= 48 && p.x <= 79 ) && ( p.y >= 151 && p.y <= 179 ) )  { key = 'S';};  // S

        if ( ( p.x >= 80 && p.x <= 111 ) && ( p.y >= 151 && p.y <= 179 ) ) { key = 'D';};  // D

        if ( ( p.x >= 112 && p.x <= 143 ) && ( p.y >= 151 && p.y <= 179 ) ) { key = 'F';};  // F

        if ( ( p.x >= 144 && p.x <= 175 ) && ( p.y >= 151 && p.y <= 179 ) ) { key = 'G';};  // G

        if ( ( p.x >= 176 && p.x <= 207 ) && ( p.y >= 151 && p.y <= 179 ) ) { key = 'H';};  // H

        if ( ( p.x >= 208 && p.x <= 239 ) && ( p.y >= 151 && p.y <= 179 ) ) { key = 'J';};  // J

        if ( ( p.x >= 240 && p.x <= 271 ) && ( p.y >= 151 && p.y <= 179 ) ) { key = 'K';};  // K
```

```
        if ( ( p.x >= 272 && p.x <= 303 ) && ( p.y >= 151 && p.y <= 179 ) ) { key =
'L';};  // L


        // Keyboard third row
```

In next if sentence, if A/a button is pushed we go back to beginning of the keyboard interface and print all characters again with lower-case character set. goto command jumps to keyboard interface before switch case.

```
        if ( ( p.x > 0 && p.x <= 42 ) && ( p.y >= 181 && p.y <= 209 ) ) {set = 1; goto
skip2;};     // uppercase/lowercase
        if ( ( p.x >= 43 && p.x <= 73 ) && ( p.y >= 181 && p.y <= 209 ) ) { key =
'Z';};    // Z
        if ( ( p.x >= 74 && p.x <= 106 ) && ( p.y >= 181 && p.y <= 209 ) ) { key =
'X';};   // X
        if ( ( p.x >= 107 && p.x <= 138 ) && ( p.y >= 181 && p.y <= 209 ) ) { key =
'C';};  // C
        if ( ( p.x >= 139 && p.x <= 170 ) && ( p.y >= 181 && p.y <= 209 ) ) { key =
'V';};  // V
        if ( ( p.x >= 170 && p.x <= 202 ) && ( p.y >= 181 && p.y <= 209 ) ) { key =
'B';};  // B
        if ( ( p.x >= 203 && p.x <= 234 ) && ( p.y >= 181 && p.y <= 209 ) ) { key =
'N';};  // N
        if ( ( p.x >= 235 && p.x <= 266 ) && ( p.y >= 181 && p.y <= 209 ) ) { key =
'M';};  // M
```

In programming language \b presents backspace or erasing character but this works on printing to pc terminal. In our case we are dealing with pixels. We cannot erase printed character just by using backspace command but we can print over it. Only “space” prints empty slot. By printing “space” on printed character we can simulate backspace.

```
        if ( ( p.x >= 267 && p.x <= 320 ) && ( p.y >= 181 && p.y <= 209 ) ) { key =
'\b';}; // backspace


        // Keyboard fourth row
```

By pushing 123 button we choose character set for numbers.

```
        if ( ( p.x > 0 && p.x <= 42 ) && ( p.y >= 211 && p.y <= 240 ) ){set = 3; goto
skip2;};       // 123
        if ( ( p.x >= 43 && p.x <= 223 ) && ( p.y >= 211 && p.y <= 240 ) ) {key = '
';};    // Space
```

```
        if ( ( p.x >= 224 && p.x <= 266 ) && ( p.y >= 211 && p.y <= 240 ) ) {key =
'.';};   // dot

        if ( ( p.x >= 267 && p.x <= 320 ) && ( p.y >= 211 && p.y <= 240 ) ) {key =
'\n';};  // enter

        break;
```

Repeat previous program and do this for lower case character set.

```
      case 1:


       tft.background( LightGrey );

       if ( ( p.x > 0 && p.x <= 31) && ( p.y >= 120 && p.y <= 149 ) )   { key = 'q';};
// q

        if ( ( p.x >= 23 && p.x <= 63 ) && ( p.y >= 120 && p.y <=149 ) )  { key =
'w';};  // w

        if ( ( p.x >= 64 && p.x <= 97 ) && ( p.y >= 120 && p.y <=149 ) )  { key =
'e';};  // e

        if ( ( p.x >= 98 && p.x <= 127 ) && ( p.y >= 120 && p.y <=149 ) ) { key =
'r';};  // r

        if ( ( p.x >= 128 && p.x <= 159 ) && ( p.y >= 120 && p.y <=149 ) ) { key =
't';};  // t

        if ( ( p.x >= 160 && p.x <= 191 ) && ( p.y >= 120 && p.y <=149 ) ) { key =
'y';};  // y

        if ( ( p.x >= 192 && p.x <= 223 ) && ( p.y >= 120 && p.y <=149 ) ) { key =
'u';};  // u

        if ( ( p.x >= 224 && p.x <= 255 ) && ( p.y >= 120 && p.y <=149 ) ) { key =
'i';};  // i

        if ( ( p.x >= 256 && p.x <= 287 ) && ( p.y >= 120 && p.y <=149 ) ) { key =
'o';};  // o

        if ( ( p.x >= 288 && p.x <= 320 ) && ( p.y >= 120 && p.y <=149 ) ) { key =
'p';};  // p


        // Keyboard second row

          if ( ( p.x >= 16 && p.x < 48 ) && ( p.y >= 151 && p.y <= 179 ) )   { key =
'a';};  // a

        if ( ( p.x >= 48 && p.x < 80 ) && ( p.y >= 151 && p.y <= 179 ) )  { key = 's';};
// s

          if ( ( p.x >= 80 && p.x < 111 ) && ( p.y >= 151 && p.y <= 179 ) ) { key =
'd';};  // d
```

```
          if ( ( p.x >= 112 && p.x < 144 ) && ( p.y >= 151 && p.y <= 179 ) ) { key =
'f';};  // f

          if ( ( p.x >= 144 && p.x < 175 ) && ( p.y >= 151 && p.y <= 179 ) ) { key =
'g';};  // g

          if ( ( p.x >= 176 && p.x < 208 ) && ( p.y >= 151 && p.y <= 179 ) ) { key =
'h';};  // h

          if ( ( p.x >= 208 && p.x < 240 ) && ( p.y >= 151 && p.y <= 179 ) ) { key =
'j';};  // j

          if ( ( p.x >= 240 && p.x < 272 ) && ( p.y >= 151 && p.y <= 179 ) ) { key =
'k';};  // k

          if ( ( p.x >= 272 && p.x < 304 ) && ( p.y >= 151 && p.y <= 179 ) ) { key =
'l';};  // l


         // Keyboard third row

          if ( ( p.x >=0 && p.x <= 42 ) && ( p.y >= 181 && p.y <= 209)){set=2; goto
skip2;};  //                                uppercase/lowercase

         if ( ( p.x >= 43 && p.x <= 73 ) && ( p.y >= 181 && p.y <=209)) { key = 'z';};
// z

         if ( ( p.x >= 74 && p.x <= 106 ) && ( p.y >= 181 && p.y <=209 ) ) { key =
'x';};   // x

         if ( ( p.x >= 107 && p.x <= 138 ) && ( p.y >= 181 && p.y <= 209 ) ) { key =
'c';};  // c

         if ( ( p.x >= 139 && p.x <= 170 ) && ( p.y >= 181 && p.y <= 209 ) ) { key =
'v';};  // v

         if ( ( p.x >= 170 && p.x <= 202 ) && ( p.y >= 181 && p.y <= 209 ) ) { key =
'b';};  // b

         if ( ( p.x >= 203 && p.x <= 234 ) && ( p.y >= 181 && p.y <= 209 ) ) { key =
'n';};  // n

         if ( ( p.x >= 235 && p.x <= 266 ) && ( p.y >= 181 && p.y <= 209 ) ) { key =
'm';};  // m

         if ( ( p.x >= 267 && p.x <= 320 ) && ( p.y >= 181 && p.y <= 209 ) ) { key =
'\b';}; // backspace


         // Keyboard fourth row

         if ( ( p.x > 0 && p.x <= 42 )  && ( p.y >= 211 && p.y <= 240 ) ) { set = 3;
goto skip2;};        // 123

         if ( ( p.x >= 43 && p.x <= 223 ) && ( p.y >= 211 && p.y <= 240 ) ) { key = '
';};    // Space
```

```
        if ( ( p.x >= 224 && p.x <= 266 ) && ( p.y >= 211 && p.y <= 240 ) ) { key = '.';};   // dot

        if ( ( p.x >= 267 && p.x <= 320 ) && ( p.y >= 211 && p.y <= 240 ) ) { key = '\n';};  // enter

      break;
```

Repeat again and do this for number character set.

```
      case 3:


       tft.background( LightGrey );

        if ( ( p.x > 0 && p.x <= 31 ) && ( p.y >= 120 && p.y <= 149 ) )   {key = '1';}; // 1

        if ( ( p.x >= 32 && p.x <= 63 ) && ( p.y >= 120 && p.y <= 149 ) )  {key = '2';};  // 2

        if ( ( p.x >= 64 && p.x <= 97 ) && ( p.y >= 120 && p.y <= 149 ) )  {key = '3';};  // 3

        if ( ( p.x >= 98 && p.x <= 127 ) && ( p.y >= 120 && p.y <= 149 ) ) {key = '4';};  // 4

        if ( ( p.x >= 128 && p.x <= 159 ) && ( p.y >= 120 && p.y <= 149 ) ) {key = '5';};  // 5

        if ( ( p.x >= 160 && p.x <= 191 ) && ( p.y >= 120 && p.y <= 149 ) ) {key = '6';};  // 6 presents backspace or erasing character but this works on printing to pc terminal. In our case we are dealing w

        if ( ( p.x >= 192 && p.x <= 223 ) && ( p.y >= 120 && p.y <= 149 ) ) {key = '7';};  // 7

        if ( ( p.x >= 224 && p.x <= 255 ) && ( p.y >= 120 && p.y <= 149 ) ) {key = '8';};  // 8

        if ( ( p.x >= 256 && p.x <= 287 ) && ( p.y >= 120 && p.y <= 149 ) ) {key = '9';};  // 9

        if ( ( p.x >= 288 && p.x <= 320 ) && ( p.y >= 120 && p.y <= 149 ) ) {key = '0';};  // 0


        // Keyboard second row

        if ( ( p.x >= 16 && p.x <= 47 ) && ( p.y >= 151 && p.y <= 179 ) )  {key = '!';};  // !

        if ( ( p.x >= 48 && p.x <= 79 ) &&  ( p.y >= 151 && p.y <= 179 ) )  {key = '#';};  // #
```

```
        if ( ( p.x >= 80 && p.x <= 111 ) && ( p.y >= 151 && p.y <= 179 ) ) { key
='%';};  // %

        if ( ( p.x >= 112 && p.x <= 143 ) && (p.y >= 151 && p.y <= 179 ) ) { key
='&';};  // &

        if ( ( p.x >= 144 && p.x <= 175 ) && (p.y >= 151 && p.y <= 179 ) ) { key
='/';};  // /

        if ( ( p.x >= 176 && p.x <= 207 ) && (p.y >= 151 && p.y <= 179 ) ) { key
='(';};  // (

        if ( ( p.x >= 208 && p.x <= 239 ) && (p.y >= 151 && p.y <= 179 ) ) { key
=')';};  // )

        if ( ( p.x >= 240 && p.x <= 271 ) && (p.y >= 151 && p.y <= 179 ) ) { key =
'=';};  // =

        if ( ( p.x >= 272 && p.x <= 303 ) && (p.y >= 151 && p.y <= 179 ) ) { key =
'?';};  // ?


        // Keyboard third row

        if ( ( p.x > 0 && p.x <= 42 ) && ( p.y >= 181 && p.y <= 209 ) ) {set = 1; goto
skip2;};     // uppercase/lowercase

        if ( ( p.x >= 43 && p.x <= 73 ) && ( p.y >= 181 && p.y <= 209 ) ) { key =
'<';};    // <

        if ( ( p.x >= 74 && p.x <= 106 ) && ( p.y >=181 && p.y <= 209 ) ) { key ='
>';};   // >

        if ( ( p.x >= 107 && p.x <= 138 ) && ( p.y >=181 && p.y <= 209 ) ) { key =
',';};  // ,

        if ( ( p.x >= 139 && p.x <= 170 ) && ( p.y >=181 && p.y <= 209 ) ) { key =
';';};  // ;

        if ( ( p.x >= 170 && p.x <= 202 ) && ( p.y >=181 && p.y <= 209 ) ) {key =
':';};  // :

        if ( ( p.x >= 203 && p.x <= 234 ) && ( p.y >=181 && p.y <= 209 ) ) {key =
'-';};  // -

        if ( ( p.x >= 235 && p.x <= 266 ) && ( p.y >=181 && p.y <= 209 ) ) {key =
'_';};  // _

        if ( ( p.x >= 267 && p.x <= 320 ) && ( p.y >=181 && p.y <= 209 ) ) {key =
'\b';}; // Backspace


        // Keyboard fourth row

        if ( ( p.x > 0 && p.x <= 42 ) && (p.y >= 211 && p.y <= 240 ) ) { set = 3; goto
skip2;};       // 123
```

```
        if ( ( p.x >= 44 && p.x <= 223 ) && (p.y >= 211 && p.y <= 240 ) ) { key = '
';};    // Space

        if ( ( p.x >= 224 && p.x <= 266 ) && (p.y >= 211 && p.y <= 240 ) ) { key =
'.';};   // Dot

        if ( ( p.x >= 267 && p.x <= 320 ) && (p.y >= 211 && p.y <= 240 ) ) { key =
'\n';};  // Enter

      break;

      };
```

Now only thing left is the chatbox engine. Building the engine can be quite difficult due of irregular character size. When we keep changing from lower-case to capital letters we print different size of characters. Different sizes equals to different amount of space from screen. To solve this dilemma we need to figure out how we can erase these irregular spaces by backspace erasing only from the area that the character required. To do this we need to store all written characters required spaces somewhere where we can check it and erase that area from right location from the screen.

Chat box engine working princible:

If chat box is pushed do nothing just loop around, if keyboard area is pushed a character is stored. If character is stored we need to check what character is it, in case it's alphabetic or numbers we store this to buffer and print this character to screen. Then we go back to wait for new action. In case of backspace being pushed we need to check the location of buffer and erase previous character from buffer and from the screen. To do this we have two arrays buffer and key_back, buffer for storing written characters and key_back to remember the location of that character on screen.

```
// Chat Box engine
if ( ( p.x > 250 && p.x <= 320 ) && ( p.y > 0 && p.y <= 119 ) ){ goto key_break; }  //
                                        OK button - exit keyboard
    if ( ( p.x > 0 && p.x <= 320) && ( p.y > 120 && p.y <= 240 ) )
    {
```

If user pushed backspace our variable (key) will have character \b. If this is the case our engine will erase the previous character.

```
    if( key == '\b' )
    {
```

If key_back = row change(\n) and variable chaty is over 23, engine will erase 23 from chaty and set variable t5 to 1; This means we have just changed row so we need to change location variable chaty to default so we can start printing from the left side of the screen. More info of variable t5 later on.

```
        if ( ( key_back[ key_i ] == '\n' ) && ( chaty > 23 ) ) { chaty = ( chaty – 23 ); t5
= 1; }
```

```
t2 = 1; t4 = 1;
```

if variable key_i is lower than 1 we set it up to default back to 1. As default variable key_i==1 start location for writing characters. Backspace array must be set to default also so we can do this by setting default value to key_back arrays slot key_i to 23. Next time we write character to screen it will be printed from p.x 23 on.

Pointers p.x and p.y presents pixel location on screen.

```
if( key_i < 1 ){ key_i = 1; key_back[ key_i ] = 23; }
key=' ';            // Empty out variable (key)
```

So now the backspace has been pushed we need to erase previous data in case t3 is not active (t3 prevents engine from repeating this before user pushed backspace again) we will decrease variable key_i(character location) and set variable t3 active. Variable x2 presents character location at buffer. Key_back is only for the screen. If variable x2 is over 0 we decrease it by 1.

```
if( t3 != 3){ t3 = 3; key_i--; }
if( x2 > 0 ){ x2--; }
}
```

End of backspace function.

If button is pushed (not backspace), variable t2 aint 1 and variable t4 is 1-> we increase key_i by 1 and set t4 to zero. This means one character will be stored to buffer and to prevent overriding previous one we increase character location by one.

```
if( t2! = 1 && t4 == 1){ key_i++; t4 = 0;}
```

If key is row change \n and variable chaty is lower than 80 , we increase chaty by 23. This 23 is the “space from screen” we use for printing every character. We store to key back this row change and increase key_i by one and set key_back[key_i] slot to 1; This means backspace function will know that row has been changed and will erase data from the right row and not from anywhere else. chaty being 23 means our row starts at y axis location 23(p.y=23).

```
if( ( key == '\n' ) && ( chaty < 80 ) ) {
chaty = ( chaty + 23);
key_back[key_i]='\n';
key_i++; key_back[ key_i ] = 1;}
```

If key is not row change(\n) but if t2 is active we will skip this function. t2 active means backspace has been pushed previously.

```
if( key != '\n' )
{
```

```
if( t2 == 1 ){goto skip;}
```

If backspace has not been pushed and user pushed keyboard area.

```
if( ( key != '\b') && ( ( p.x > 0 ) && ( p.x <= 320 ) ) && ( ( p.y >= 120 ) && ( p.y <= 240 ) ) )
{
```

If set is 1 or 3 character set is either lower case or number then space taken from screen is set to 20. Here engine will add taken space 20 to previous location value. Which mean if previous location was 73 now it is 93. This location presents p.x(x axis). If set is 2 character set uppercase then space taken is 22.

```
if( set == 1 || set == 3) { key_back[ key_i ] = ( key_back[ key_i - 1 ] + 20 ) ; }
if( set == 2 ) { key_back[ key_i ] = ( key_back[ key_i - 1 ] + 22 ) ; }
```

If x axis location value is over 230 and chaty is smaller than 80 chaty is increased by 23 and row change will be added to key_back array. We also increase the key_i here to prevent next character overriding this action. Also key_back is set to 23 default start location for character. chaty=y axis.

```
if( ( key_back[ key_i ] > 230 ) && ( chaty < 80 ) ) { chaty = ( chaty + 23 ); key_back[ key_i ] = '\n'; key_i++ ; key_back[ key_i ] = 23; }
}
}
skip:
```

Here we store key to buffer array into location of x2 variable.

```
buffer[ x2 ] = key;
```

If chaty is over 80 and key_back is over 200 and t2 aint 1. This means we are at end of the chat box and we have no more space for any character. We want to show this for the user that it’s the end so engine will print to screen star. Depending on chosen character set we backspace by one character.

```
if( chaty > 80 && key_back[ key_i ] >= 200 && t2 != 1 )
{
key='*';

if(set == 1 ||set == 3) { key_back[key_i]=(key_back[key_i-1]-20);}
if(set == 2) { key_back[key_i]=(key_back[key_i-1]-22);}
key_i--;

}
```

If key_back is lower than 200 we don’t skip, Meaning every time we reach over 200 on x axis means row change has to happen and when ron change happens we don’t want to print character \n to screen.

```
if( key_back[ key_i ] < 200 ){ key_skip = 0;}
```

If t5 aint active engine will print to tft.locate(x-axis,y-axis) pushed buttons character.

```
if(t5 != 1 ) {
   tft.locate( key_back[ key_i ], chaty );
   tft.putc(key);
```

If key aint star and key aint row change engine will store variable key character to buffer.

```
   if(key!='*' && key!='\n') { buffer[x2]=key; }
```

If key is row change, engine will store empty slot to buffer.

```
   if(key=='\n') { buffer[x2]=' '; }
  }
```

Set t5 to 0.

```
t5=0;
```

If t2 is active engine will decrease key_i by one.

```
//ADDS & SUBS
if(t2==1)
  {
   key_i--;
  }
```

If t2 is not active engine will increase key_i by one, also if key aint star we increase buffer location by 1

```
if(t2!=1){
   key_i++;
   t3=0;
   if(key!='*') { x2++; }
   }
t2=0;
```

Engine will wait for 0.1 second to give some time for user to remove hand or pen from the button before taking in new action.

```
wait(0.1);
```

```
    };
```

```
 key_break:
```

We increase buffer location by 1 and store there comma. This means that user pushed OK button and we are ready to finish. We store comma to the buffer to know where does previously written string end.

```
 x2++; buffer[x2]=',';    Loading image to Mikroe TFT-Proto
};

// Congraz keyboard done
```

Tips!:

Always keep in mind that every code takes time and doing multiple operations with microprocessor while also controlling screen is heavy duty. For faster operations is better to have one microprocessor as “screen control” and one for other operations. Exmpl if drawing rectangle or triangle, instead of drawing it with lines or pixel by pixel draw it as rectangle or triangle. Hint! using black as background will consume less current and is faster to process.

1.1.5 Loading image to Mikroe TFT-Proto

1. Download GIMP or any image editing software that allows scaling image. http://www.gimp.org/
2. Open the picture you want to load with GIMP. Example image: http://www.wallhoster.com/wp-content/uploads/3D-Animals-wallpapers.jpg
3. Scale image to size you want. Max 320*240 this is resolution of Mikroe TFT-Proto. Push Shift+T at GIMP width 320 and height 240, then push scale.
4. Image-> Canvas Size/Set 320*240.
5. Save image as test.bmp to mbed. Export As ->Select File Type/Windows BMB-> Additional/16byte-R5G6B5.
6. In Code us the BMP_16 command to draw image from local drive.

   BMP(Start pixel x, Start pixel y, “file_name.bmp”);

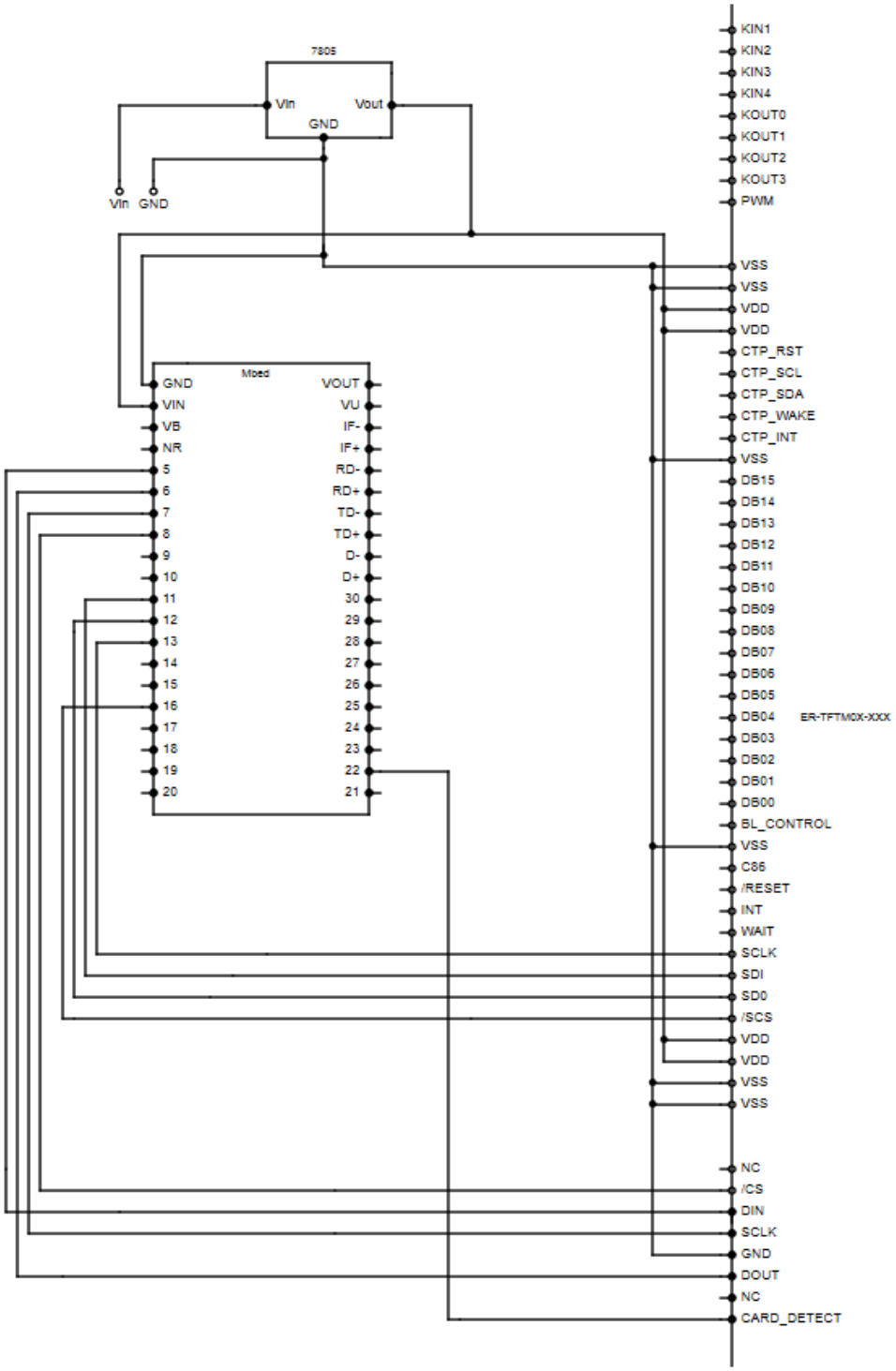
Example code:

```
#include "mbed.h"
```

```
 1
// TFT files
#include "SPI_TFT_ILI9341.h"
#include "Arial12x12.h"
#include "Arial24x23.h"
#include "Arial28x28.h"
#include "font_big.h"
#include "touch_tft.h"
// TFT pins
Serial pc(USBTX, USBRX);
touch_tft ttp19,p18,p17,p16,p11, p12, p13, p14, p20, p22,"TFT"); // x+,x-,y+,y-,mosi,
miso, sclk, cs, reset, dc
DigitalOut myled(LED1);


int main()
{
 //TFT initialize
    tft.claim(stdout);      // send stdout to the TFT display
    tft.set_orientation(1);
    tft.background(Black);    // set background to black
    tft.foreground(White);    // set chars to white
    tft.cls();               // clear the screen


    // Image call function
    int err = tft.BMP_16(0,0,"test.bmp");             // Print image
    if (err != 1) pc.printf(" - Err: %d",err);             // if file not found print error to
pc(teraterm / putty)
}
```

1.2 ER-TFT070, 7'' TFT-touch

Kuva 3: RA8875 Test Board



ER-TFT070 is RA8875 driven display module from EastRising(buydisplay), it includes resistive touch panel, sd-slot, tft-display, and internal backlight control. Due to its 7''

screen it can be made very user-friendly with decent brightness. Display has 800 * 480 pixels and is fairly cheap. Library can be found from Mbed website but needs slight modification. Modified library and test board layout is included wat ReskiT zip.

Datasheet can be found from http://www.buydisplay.com/download/manual/ER-TFTM070-5_Datasheet.pdf Display can be bought from www.buydisplay.com or www.aliexpress..com

These display modules are great if there is need for different size of screen with same software.

Configuring ER-TFT070

Before getting into programming (depends on the case) the ER-TFT070 module needs some altering. In this case we are using 4 – wire Serial SPI communication, resistive touch panel, 5V supply. Intsruction for configurating the board are at datasheet page 11. ”J” Represents resitors/ Connections at back of the module.

For this configuration J8, J9, J11, J13, J15 open and J10, J12, J14, J16 short. R1 – R3 replace with 10K resistor, R35 – R38 with 0R resistor, and leave R39 – R42 open. In case 3.3 V supply is used → short J8.

Library modifications

Modification done for RA8875 original library(David Smart, http://mbed.org/users/WiredHome/code/RA8875/) :

Original library was made for 480 x 360, which means it was necessary to convert to 800 * 480.

1. In RA8875.cpp at line 27 and 28: Change display width to 800 and display height to 480.
2. In RA8875.cpp at line 79 change x to 800 (width() >= x) and height() >= x to 480.
3. Replace the TouchPanelRead with following:

```
unsigned char RA8875::TouchPanelRead(loc_t* x, loc_t* y)
{
   unsigned char touchready;
   static int xbuf[TPBUFSIZE], ybuf[TPBUFSIZE], sample = 0;
   int i, j, temp;



   if( (ReadCommand(INTC2) & RA8875_INT_TP) ) {        // Test for TP Interrupt pending in register INTC2
      // Get the next data samples
```

```
      ybuf[sample] =  ReadCommand(TPYH) << 2 | ( (ReadCommand(TPXYL) & 0xC)
>> 2 );   // D[9:2] from reg TPYH, D[1:0] from reg TPXYL[3:2]
      xbuf[sample] =  ReadCommand(TPXH) << 2 | ( (ReadCommand(TPXYL) & 0x3)
);   // D[9:2] from reg TPXH, D[1:0] from reg TPXYL[1:0]


      // Check for a complete set
      if(++sample == TPBUFSIZE) {


        // Sort the Y buffer using an Insertion Sort
         for(i = 1; i <= TPBUFSIZE; i++) {
            temp = ybuf[i];
            j = i;
            while( j && (ybuf[j-1] > temp) ) {
               ybuf[j] = ybuf[j-1];
               j = j-1;
            }
            ybuf[j] = temp;
         } // End of Y sort
         // Sort the X buffer the same way
         for(i = 1; i <= TPBUFSIZE; i++) {
            temp = xbuf[i];
            j = i;
            while( j && (xbuf[j-1] > temp) ) {
               xbuf[j] = xbuf[j-1];
               j = j-1;
            }
            xbuf[j] = temp;
         } // End of X sort
         // Average the middle half of the  Y values and report them
         j = 0;
         for(i = (TPBUFSIZE/4) - 1; i < TPBUFSIZE - TPBUFSIZE/4; i++ ) {
            j += ybuf[i];
         }

         int16_t tresy,tresx;
         tresy = j * (float)2/TPBUFSIZE;    // This is the average
```

```
        // Average the middle half of the  X values and report them

        //yyx = j * (float)2/TPBUFSIZE;    // This is the average
        // Average the middle half of the  X values and report them

        j = 0;
        for(i = (TPBUFSIZE/4) - 1; i < TPBUFSIZE - TPBUFSIZE/4; i++ ) {
            j += xbuf[i];
        }
        //xxy = j * (float)2/TPBUFSIZE;
        tresx = j * (float)2/TPBUFSIZE;    // This is the average
        // Tidy up and return

        // convert to pixels

        tresx = ((float)resolutionx/1111) * tresx;
        tresy = ((float)resolutiony/1051) * tresy;

        // offset add
        if(tresx > 392.5)
        {
            *x = (float)(tresx - ((20/(0.5 * tresx)) *0.5*tresx));
        }
        if(tresx < 392.5)
        {
            *x = (float)(tresx - ((55/(0.5*tresx)) * (0.5*tresx)));
        }

        if(tresy > 212.5)
        {
            *y = (float)(tresy - ((1/(0.5 * tresy)) *0.5*tresy));
        }
        if(tresy < 212.5)
        {
            *y = (float)(tresy - ((56/(0.5*tresy)) * (0.5*tresy)));
```

```
        }


        touchready = 1;
        sample = 0;             // Ready to start on the next set of data samples
        //pc.printf("\n Touchready: %d:%d", yyx,xxy);
      }
      else {
        // Buffer not yet full, so do not return any results yet
        touchready = 0;
      }


      WriteCommand(INTC2, RA8875_INT_TP);            // reg INTC2: Clear that TP
interrupt flag
   } // End of initial if -- data has been read and processed
   else
   {
      touchready = 0;         // Touch Panel "Int" was not set
      return touchready;
    }
}
```

Reading touch values needed some modifying for more accurate display pixel ↔ touch pixel interfacing.

Programming first interface. Ready made library can be found from Code.zip

```
// Definations
#Include ”mbed.h”
#include ”RA8875.h”
// Fonts
#include "Arial12x12.h"
#include "Arial24x23.h"
#include "Arial28x28.h"ER-TFT070, 7'' TFT-touch
#include "font_big.h"

// Variables
```

```
 int16_t tx,ty;
// Define pins
RA8875 lcd(p11, p12, p13, p16, NC, "tft");    // MOSI, MISO, SCK, /ChipSelect, /reset, name
Serial pc(USBTX, USBRX);               // Debugging with teraterm

void LCD_Initialize()
{
 pc.baud(9600);
 pc.printf(”\n\r LCD initializing”);
 lcd.Power(true);          // Turn display on
 lcd.Backlight(1.0);       // Set backlight power
 lcd.TouchPanelInit();     // Initialize touch panel
}
// Touch controll
void UI::touch()
 {
   while(1)
    {
       tx=0;ty=0;

       if (lcd.TouchPanelRead(&tx, &ty)== true) // read analog pos.
       {
          pcui.printf("\n\r Touch 1 Point x:%d, y:%d", tx, ty);
          }
          wait_ms(30);
          break;
      }
           else
           { MBED – Textlists with automated touch action slots
              tx=0;ty=0;
           }
      }
 }
```

```
// Folder Icon
void Folder_Icon(float x, float y, char Color)
{
   /*  Folder*/
   // Folder Frame
   switch(Color)
   {
      case 'w':


         lcd.line(((x - 0.04) * scrx),((y - 0.0525) * scry), ((x - 0.05) * scrx),((y - 0.0275) * scry),White);   // left corner
         lcd.fillrect(((x - 0.04) * scrx),((y - 0.0525) * scry),( (x + 0.04) * scrx),( (y - 0.0275) * scry), White); // Upper frame
         lcd.line(((x + 0.04) * scrx),((y - 0.0525) * scry), ((x + 0.05) * scrx),((y - 0.0275) * scry),White);   // Right corner


         lcd.fillrect(((x - 0.05) * scrx),((y - 0.0275) * scry),( (x + 0.05) * scrx),( (y + 0.0525) * scry), White); // Lower frame


         // Hatch
         lcd.line(((x - 0.05) * scrx),((y - 0.0275) * scry), ((x - 0.015) * scrx),((y - 0.0275) * scry),Black);


         lcd.line(((x - 0.015) * scrx),((y - 0.0275) * scry), ((x - 0.01)* scrx),((y - 0.02) * scry),Black);
         lcd.line(((x - 0.01) * scrx),((y - 0.02) * scry), ((x + 0.015) * scrx),((y - 0.02) * scry),Black);
         lcd.line(((x + 0.015) * scrx),((y - 0.02) * scry), ((x + 0.02 ) * scrx),((y - 0.0275) * scry),Black);
         lcd.line(((x + 0.02) * scrx),((y - 0.0275) * scry), ((x + 0.05) * scrx),((y - 0.0275) * scry),Black);


         lcd.set_font((unsigned char*) Arial24x23);
         lcd.background(White);
         lcd.foreground(Black);
         lcd.puts(((x - 0.042) * scrx),((y + 0.0025) * scry),"Tools");
         /****/
      break;
```

```
        case 'g':
            lcd.line(((x - 0.04) * scrx),((y - 0.0525) * scry), ((x - 0.05) * scrx),((y - 0.0275) * scry), Green);   // left corner
            lcd.fillrect(((x - 0.04) * scrx),((y - 0.0525) * scry),( (x + 0.04) * scrx),( (y - 0.0275) * scry), Green); // Upper frame
            lcd.line(((x + 0.04) * scrx),((y - 0.0525) * scry), ((x + 0.05) * scrx),((y - 0.0275) * scry), Green);   // Right corner


            lcd.fillrect(((x - 0.05) * scrx),((y - 0.0275) * scry),( (x + 0.05) * scrx),( (y + 0.0525) * scry), Green); // Lower frame


            // Hatch
            lcd.line(((x - 0.05) * scrx),((y - 0.0275) * scry), ((x - 0.015) * scrx),((y - 0.0275) * scry),Black);


            lcd.line(((x - 0.015) * scrx),((y - 0.0275) * scry), ((x - 0.01)* scrx),((y - 0.02) * scry),Black);
            lcd.line(((x - 0.01) * scrx),((y - 0.02) * scry), ((x + 0.015) * scrx),((y - 0.02) * scry),Black);
            lcd.line(((x + 0.015) * scrx),((y - 0.02) * scry), ((x + 0.02 ) * scrx),((y - 0.0275) * scry),Black);
            lcd.line(((x + 0.02) * scrx),((y - 0.0275) * scry), ((x + 0.05) * scrx),((y - 0.0275) * scry),Black);


            lcd.set_font((unsigned char*) Arial24x23);
            lcd.background(Green);
            lcd.foreground(Black);
            lcd.puts(((x - 0.042) * scrx),((y + 0.0025) * scry),"Tools");
            /****/
        break;
        default:
        break;
    }
}

// Starting interface
int main()
{
```

```
 lcd.background(Black); // Set background color
 lcd.foreground(White); // Set foreground color
 lcd.cls();    // Clear screen

// Printing text
 lcd.set_font((unsigned char*) Arial24x23);
 lcd.puts((0.1 * scrx),(0.2 * scry), "Welcome, Font 24X23 ");
 lcd.set_font((unsigned char*) Arial28x28);
 lcd.puts((0.1 * scrx),(0.4 * scry), "Welcome, Font 28X28 ");
 lcd.set_font((unsigned char*) Neu42x35);
 lcd.puts((0.1 * scrx ),(0.6 * scry ),"Welcome, Font 42X35");
 wait(10);   // wait 10 seconds


 lcd.cls();    // Clear screen
}
```

1.2.1 MBED – Textlists with automated touch action slots

These functions will draw a square textlist box with a slot for each text of inputted strings. It also adds a action slot/button around the text.

```
Variables:
float TextListLeftBorder;
float TextListRightBorder;
float TextListTopBorder;
float TextListGap;
float TextListSlots;
char TextListEnable = 0;


void TextListSet(float LeftBorder, float RightBorder, float TopBorder, float Gap_, int
Slots_)
{
    TextListLeftBorder = LeftBorder;
    TextListRightBorder = RightBorder;
    TextListTopBorder = TopBorder;
```

```
   TextListGap = Gap_;
   TextListSlots = Slots_;
}

void TextListDrawInt(int Multiples, char ListBuffer[][10])
{
   lcd.set_font((unsigned char*) Arial28x28);
   for(int i = 1; i <= Multiples; i++)
  {
     sprintf(buffer, "%d", ListBuffer[1][i]);
      lcd.puts(TextListLeftBorder, (TextListTopBorder +( (i - 1) * TextListGap)), buffer);
   }
}

void TextListDraw1(int Multiples, char ListBuffer[][10])
{
   TextListEnable = 1;
   lcd.set_font((unsigned char*) Arial28x28);
   for(int i = 1; i <= Multiples; i++)
   {
      sprintf(buffer, "%s", ListBuffer[i]);
      lcd.puts(TextListLeftBorder, (TextListTopBorder +( (i - 1) * TextListGap)), buffer);
     }
}

void TextListDraw2(int Multiples, char ListBuffer[][10], char ListBuffer2[][10])
{
  lcd.set_font((unsigned char*) Arial28x28);
  TextListEnable = 1;

  for(int i = 1; i <= Multiples; i++)
  {
     sprintf(buffer, "%s", ListBuffer[i]);
     lcd.puts(TextListLeftBorder, (TextListTopBorder+( (i - 1) * TextListGap)), buffer);
     sprintf(buffer,"%s", ListBuffer2[i]);
```

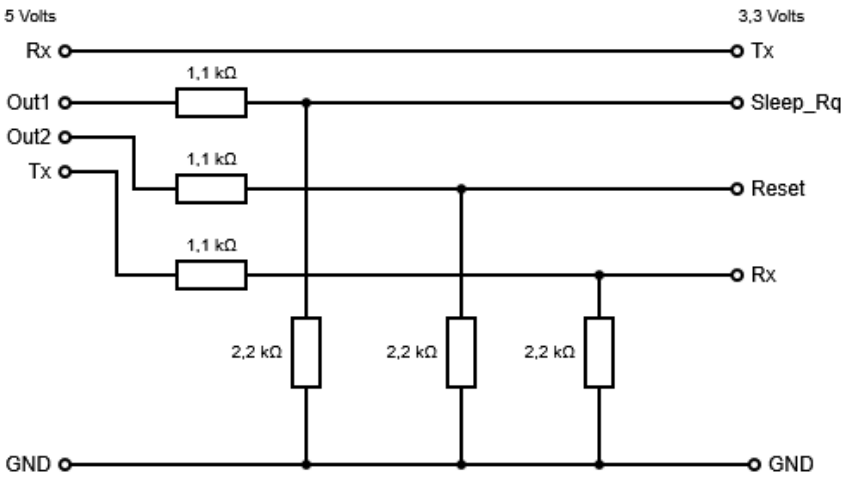
```
    lcd.printf(" %s", buffer);
  }
}
int TextListSlotTouch()
{
  if(TextListEnable == 1)
  {
   if( ((TouchPoint.x > (TextListLeftBorder - 0.02 * scrx)) && (TouchPoint.x <
   (TextListRightBorder + (0.09 * scrx) ))) && TouchPoint.y > (TextListTopBorder -
   TextListGap)  && TouchPoint.y < ((TextListSlots * TextListGap) + TextListTopBorder))
    {
      TextListActive = 1;
      for(int i_ = 0; i_ < (TextListSlots + 1); i_++)
      {
        if(( (i_ * TextListGap) + (TextListTopBorder - TextListGap)) > TouchPoint.y)
        {
          if(TextListTouchActive[i_] < 1)
          {
            TextListTouchActive[i_] = 1;
          }
          else
          {
            TextListTouchActive[i_] = 0;
          }
          return i_;
        }
      }
    }
  }
  return 0;
}

void TextListSlotTouchAction(int Slot, char Selector)
{
if(Slot >= 1 && TextListActive == 1)
{
```

```
if(TextListTouchActive[Slot] == 1)
{
lcd.foreground(Pink);
lcd.background(Gray);
TextListSlotUpdate(Slot);
if(Selector == 1)
{
TextListEnable = 0;
TextList2Enable = 1;

 TextList2Set(TextListLeftBorder - (0.15 * scrx), TextListLeftBorder - (0.05 * scrx),
TouchPoint.y, TextListGap, ts.Rider_A);
lcd.set_font((unsigned char*) Arial28x28);
TextList2Draw1(ts.Rider_A, ts.DB_.FirstName, 1 );
}
}
else if(TextListTouchActive[Slot] == 0)
{
lcd.foreground(Black);
lcd.background(Gray);
TextListSlotUpdate(Slot);
}

lcd.foreground(Black);
lcd.background(Gray);
}
}

void TextListSlot1(int Slot_, char ListBuffer[])
{
   sprintf(buffer, "%s", ListBuffer);
   lcd.set_font((unsigned char*) Arial28x28);
   lcd.puts(TextListLeftBorder, (TextListTopBorder +( (Slot_ - 1) * TextListGap)), buffer);
}

void TextListSlot2(int Slot_, char ListBuffer[], char ListBuffer2[])
```

```
{
   lcd.set_font((unsigned char*) Arial28x28);
   sprintf(buffer, "%s", ListBuffer);
   lcd.puts(TextListLeftBorder, (TextListTopBorder+( (Slot_ - 1) * TextListGap)), buffer);
   sprintf(buffer,"%s", ListBuffer2);
   lcd.printf(" %s", buffer);
}
```

1.3 XBEE

Getting into xbee can take some time but will prove to be worth it. It's easy way to implement wireless communication for your application. Xbee operates on 3.3 volts so if your microcontroller works on higher voltage using voltage divider is necessary.



Sometimes its necesssary to empty xbee buffer, this can be done with simple code.

Code for Xbee flushing:

```
char flush;
 if(xbee_.readable()==1)              // If xbee has data, xbee is readable
```

```
{
        do
        {                       // Get character from xbee until it’s not readable
                flush=xbee_.getc();     // Copy xbee data to trash
        }
        while(xbee_.readable()==1);
}
```

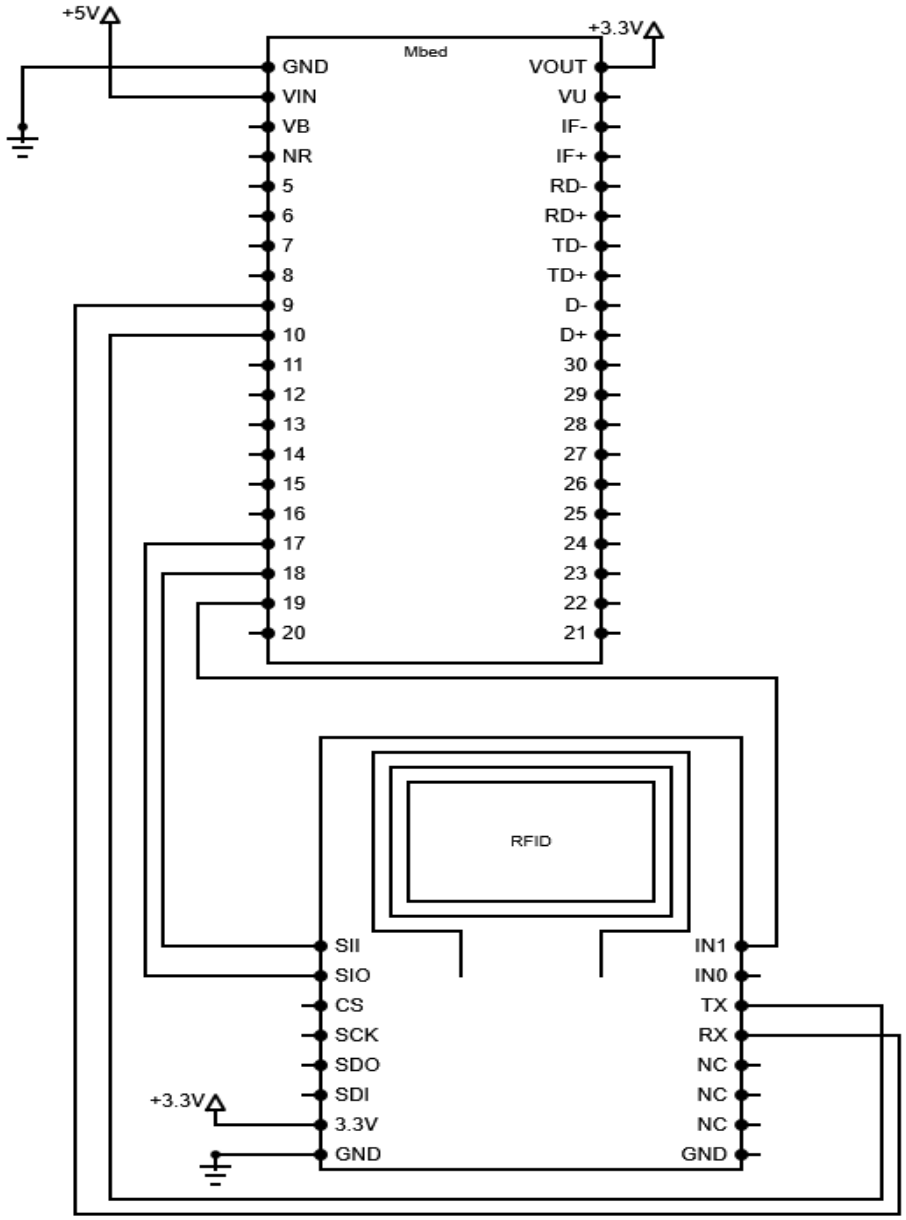
This code will check if there is any data incoming from xbee to your microcontroller(tested on mbed). This will read the uart data until the xbee’s buffer is empty.

1.4 RFID

1.4.1 RFID Click

Rfid Click is rfid reader/writer from MIKROE. Rfic Click operates on 13.56MHz and works on 3.3V, ideal for mbed. http://www.mikroe.com/click/rfid/. Libraries included in Code.zip. Before plugging RFID CLICK requires some tinkering. 0 value resistors has to be soldered to position B to use UART(Default A, SPI).

Proper instruction for soldering are at mikroe website.



Example code for using RFID Click. Some of the code is ported from [11].(Source)

```
#include "mbed.h"
#include "CR95H.h"
#include "xbee.h"

CR95H rf(p9,p10,p19,p17,p18); // Pins UART TX, RX, IRQ, IS0, IS1
Serial pc(USBTX,USBRX);              // For debugging
```

```
int main()
{
 pc.printf("\n\r Setting Database");

// Next function will store users to database. Set_DataBase(Slot, ”User”, ”RFID tag code”);
// Database tag code will be compared to receifed RFID tag code if tag codes matches user has been    //detected
    rf.Set_DataBase(1,"User1","213351726900");
    rf.Set_DataBase(2,"User2","221502499200");
    rf.Set_DataBase(3,"User3","1651842313000");

    pc.printf("\n\r Starting RFID\n\r");                // Print to pc terminal
    wait(1);
    rf.Start();// Initialize rfid

    // TEst xbee

    while(1)
    {
        rf.User = 0;          // Reset detected user
        rf.Detected = 0;    // Reset Detect flag
        rf.Sense_Tag();      // Sense for tag, if tag found set Detect flag high. After detecting tag, tag code i
                             // is compared to UserDB tag  codes if match is found, rf.User stores detected                          // slot value
        if(rf.Detected == 1)
        {
            sprintf(Data," %s!\n\r",rf.UserDB[rf.User]);              // Copy data  from UserDB[slot] to array Data
            wait_ms(200);
            pc.printf(Data);

            rf.Detected = 0;
        }
        //pc.printf("\n\r Detecting user");
```

```
   }
}
```

1.4.2 RDM6300

RDM6300 is inexpensive rfid tag reader. It works similiar to RFID Click except its only capable of reading 125khz tags or cards. Programming RDM6300 is more challenging than RFID Click because of its reading speed. RFID Click recognizes when rfid card is read and moves it to buffer. RDM6300 keeps reading the tag or card continously without any recognizable markings. To recognize read RFID tag we create a program that stores RFID card code to buffer when two simillar codes is read.

```
#include ”mbed.h”
Serial _rfid();
DigitalOut _on;
Timer Debouncer;
char Reference[15];
char Buffer[30];
char TagID[15];
 unsigned short sdata[18];
 unsigned short rdata[18];
 unsigned short res, dataNum;
 unsigned short j, tmp;

int x_pos;
int x_pos_old;
char CR95HF_ID[13];
char ID[20];
char ID_old[20];
char txt_hex[3];
char flag;
int dsize;
char TagDB[15][20]; // user database
char UserDB[15][20];
char RFID_flag;
// Set baudrate
void RFID_Baudrate(int rate)
{
  _rfid.baud(rate);
}

char ReadTag(int TagSize)
{
   int Check = 0;

   int d = 0;
```

```
for(int i = 0; i <= 15; i++)
{
   TagID[i] = 0;
   Reference[i] = 0;
}

for(int i = 0; i <= TagSize; i++)
{
   if(Buffer[i] != ' ')
   {
      Reference[i] = Buffer[i];
   }
}

for(int i = (TagSize); i <= (2*TagSize)+2; i++)
{
   if(Buffer[i+1] != ' ')
   {
      TagID[i-TagSize] = Buffer[i+1];
   }
}
// Confirm
int dstart = 0;
redo:

for(int i = 0; i <= 15; i++)
{

   if(Reference[d] == TagID[i])
   {
      d++;
      Check++;
   }
   else
   {
      d  = dstart;
   }

   //pc2.printf("\n\rCheck %d  %d::%d, %d",dstart, Reference[d], TagID[i], Check);

}

if(Check >= TagSize)
{
   pc2.printf("\n\r Success\n\r");
   return 1;
}
```

```
   else if(dstart == 3)
   {
      //pc2.printf("\n\r Fail");
      return 0;
   }
   else
   {
      dstart++;
      Check = 0;
      goto redo;
   }
   /*
   else
   {
      for(int i = (TagSize - 1); i <= ((2 * TagSize) - 1); i++)
      {
         TagRef[d] = Buffer[i];
         d++;
      }
      d = 0;
      for(int i = TagSize; i <= ((3 * TagSize) - 1); i++)
      {
         TagID[d] = Buffer[i];
         d++;
      }
      d = 0;

      // Confirm

      for(int i = 0; i <= TagSize; i++)
      {
         TagRef[i] = Buffer[i];
         Check++;
      }

      if(Check == TagSize)
      {
         return 2;
      }
      else
      {
         return 0;
      }

   }
   */

}
```

```
// Scan Function. This function polls the RDM6300 module if card is whiped over, card data is stored to buffer
char Scan()
{
    int d = 0;
    // Clear RFID buffer
    for(int i = 0; i <= 30; i++)
    {
        Buffer[i] = 0;
    }

// If RDM6300 module is available for reading, start polling.

    if(_rfid.readable() == 1)
    {
        Debouncer.reset();
        Debouncer.start();
        // Poll for 1 second. If 1 second is exceeded stop polling.
        // This prevents mbed getting stuck.
        // Variable d is increased by one everytime character is found from RDM6300
        while(Debouncer.read() < 1)
        {
            if(_rfid.readable() == 1 && d < 30)
            {
                Buffer[d] = _rfid.getc();
                d++;
            }
        }
        Debouncer.stop();
        Debouncer.reset();
    }
    //Here the tag id length is 12. If d variable is higher than 0 and Tag was detected from buffer
   // return 1. This indicates that tag was detected and read correctly. In case of fail event return 0.
    if( (d > 0) && (ReadTag(12) == 1) )
    {
        pc2.printf("\n\r Tag ID IS: %s", TagID);
        return 1;
    }
    else
    {
        return 0;
    }

}
```

```
char Sense_Tag(int TagSize)
{
   if (Scan() == 1)
   {
      for (int i=0; i<= 15; i++)
      {
         ID[i] = TagID[i];
      }

      int Check = 0;

      for (int i = 0; i <= 15; i++)
      {
         if(ID[i] == ID_old[i])
         {
            Check++;
         }
      }

      if(Check < TagSize)
      {
         pc2.printf("\n\r New ID is %s, %d", ID, Check);
         pc2.printf("\n\r Old ID is %s, %d", ID_old, Check);

         for(int i = 0; i <= (TagSize+1); i++)
         {
            ID_old[i] = ID[i];
         }
         return 1;
      }
      else if(Check >= TagSize)
      {
         return 2;
      }
   }
   return 0;
}
void Set_DataBase(int slot, char Name[20], char TagID[20]);
char DetectUser();

int main(void)
{
  return 0;
}
```

2 Teensy++

2.1 Required equipment/tools

Programmer: Teensy loader, http://www.pjrc.com/teensy/loader.html

Compiler: Atmel Studio 6, http://www.atmel.com/tools/atmelstudio.aspx

Microcontroller: Teensy module, https://www.pjrc.com/store/teensypp.html

Datasheet: AT90USB64/128, http://www.atmel.com/Images/doc7593.pdf

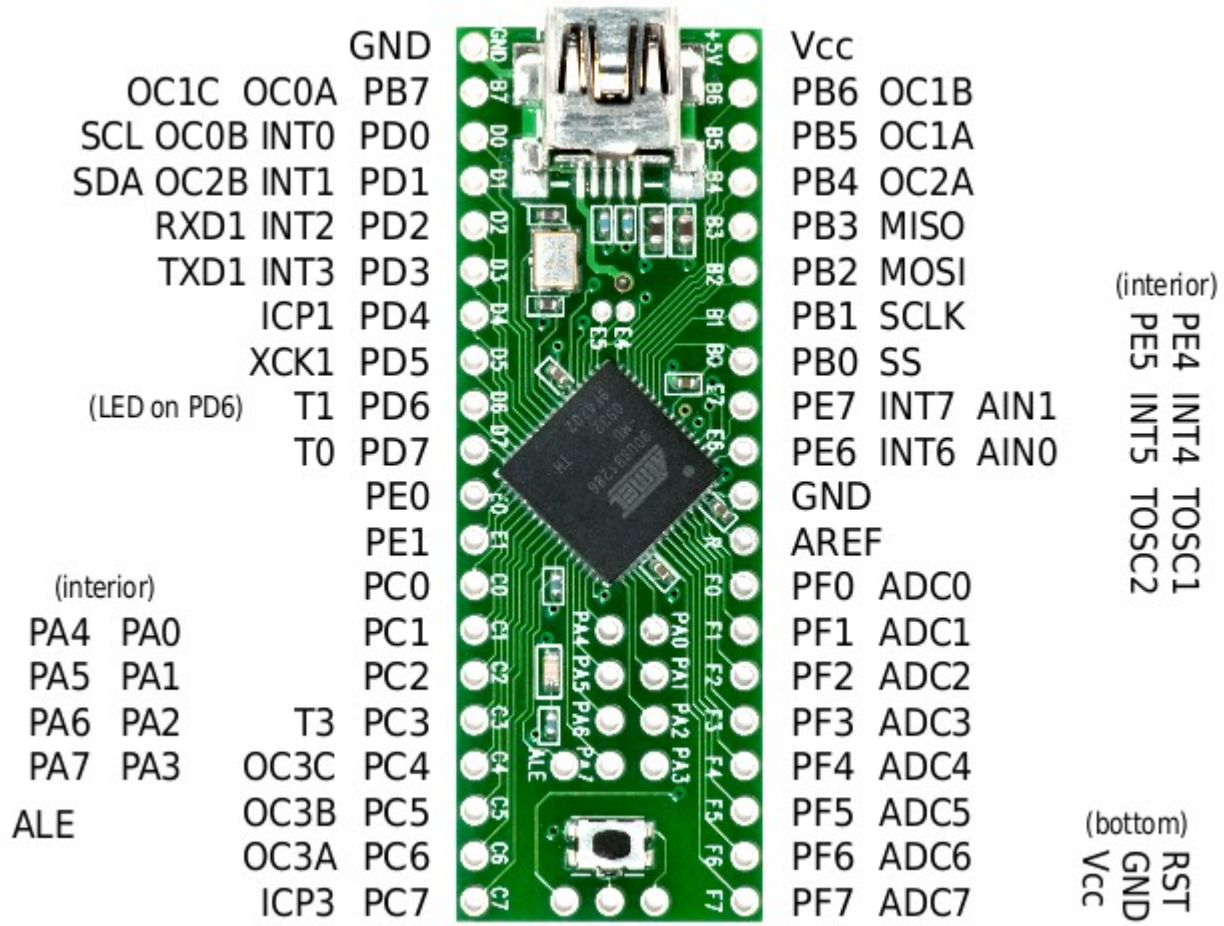


Illustration 4: Teensy

2.2 Setting up project

We assume that atmel studio 6 is used.

Open atmel studio 6 from start programs.

Click New Project

Select GCC Executable Project, name it and click ok

Select your device and click ok. I’m using AT90USB646. Device name is at top of microcontroller chip.

On solution explorer you will have all your project files.

2.3 Interrupt Timer

In here we use 16 byte time TCNT3. To make the timer easier to controll we make it into couple different function. First function we call as TickerInit, second TickerStart, third TickerReset and last TickerStop.

Interrupt Timer can be used as a timer or to delay program operations without putting the whole program in halt.

TickerInit function initalizes the Ticker for 1ms operation. Open at90usb1286 datasheet and go to page 136 16-bit Timer/Counter

We set prescaler to 8 which means that on 8Mhz clock speed timer overflow happens on 1000000 counts. Timer3 is 16 bit timer so 2^16 equals to 65535. At 8MHz one clock cycle is 1/(8*10^6) = 125ns/cycle but with prescaler it's (1/1*10^6) = 1us. With 256 prescaler one cycle is 1us. To get 1ms we need to divide 1ms with the cycles = 1ms/1us => (1*10^-3)/(1*10^-6) = 1000 cycles.

```
#define On 1            // Defines word On as 1
#define Off 0           // Defines word Off as 0
```

volatile short int ticker = 0;

volatile char TickerEnable = Off;

```
void TickerInit()
{
  TCCR3A = 0;           // Set everything to normal operation
  TCCR3B = (1 << CS11);  // Set prescaler to 8
  Ticker = 0;  // reset ticker
  TickerEnable = Off;  // initially ticker is off
  TIMSK3 = (1 << TOIE3); // Enable Timer
  TCNT3 = (65536 - 1000); // 1000 cycles equal to 1ms, TCNT3 overflows when it counts over                                  }                                  // 65536. This causes interrupt
```

Function TickerStart is used to start the ticker, it sets TickerEnable variable to 1 which allows in isr vector ticker variable counting. It resets the ticker variable.

```
void TickerStart()
{
  TickerEnable = On;
  Ticker = 0;
}
```

Function TickerStop is used to stop the ticker, it sets TickerEnable to 0. It also resets the ticker.

```
void TickerStop ()
{
  TickerEnable = Off;
  Ticker = 0;
}
```

Function TickerReset resets the ticker.

```
void TickerReset()
{
  Ticker = 0;
}
```

when overflow happens in Timer3 ISR vector is called. From datasheet we can see that Timer3 isr vector is called as TIMER3_OVF_vect.

```
ISR(TIMER3_OVF_vect)
{
  TCNT3  =  (65536 - 1000); // 1000 cycles equal to 1ms
  if (TickerEnable == On)
  {
    Ticker++;
  }
}
```

2.4 UART communication

```
// CPU clock
  #define F_CPU 8000000UL


  #include <avr/io.h>
  #include <util/delay.h>
```

```
#include <avr/sleep.h>
#include <avr/interrupt.h>
#include <inttypes.h>
#include <stdint.h>

using namespace std;

      /**UART**/ // Uart code reference taken from [7].
#define BAUD_RATE 9600           // Set baudrate
#define UBRRVAL ((F_CPU/(16UL*BAUD_RATE))-1)

#define Serial_DDR DDRD // Serial ddrd
#define Serial_PORT PORTD // Serial port
#define Serial_ PIND // Serial pins
#define TX PD3       // Serial din
#define RX PD2       // Serial dout

volatile char data;
char Datab[100];
char buffer[100];

typedef uint16_t u16;


        // initialize UART

void UA_initilize()
{
        CPU_PRESCALE(CPU_8MHz); // set cpu frequency for teensy

        UBRR1H = (UBRRVAL >> 8); //low byte
        UBRR1L = (uint8_t)UBRRVAL; //high byte
        UCSR1B |= (1<<TXEN1)|(1<<RXEN1); // Enable Tx and Rx interrupt

//Set data frame format: asynchronous mode,no parity, 1 stop bit, 8 bit size //Enable
Transmitter and Receiver and Interrupt on receive complete
        UCSR1C = (0<<UMSEL11) | (0<<UMSEL10) | (0<<UPM11) | (0<<UPM10) |
(0<<USBS1) |        (0<<UCSZ12) | (1<<UCSZ11) | (1<<UCSZ10);

        SERIAL_DDR = 0xFF;
        SERIAL_ = 0x00; // Gate D pins to zero

}

uint8_t USART_vReceiveByte(void)
{
        // Wait until a byte has been received

        while((UCSR1A & (1 << RXC1)) == 0);
        // Return received data
        return UDR1;
}

        // Write character to UDR(Serial buffer)
void UA_write(char data)
{
     while (!(UCSR1A & (1 << UDRE1))); // as long UDRE ain't empty do nothing
        UDR1=data; // write to UART buffer, send serial data
}

char UA_wc(char wcData)
{
        char success = 0;
```

```
        // Wait for transmission buffer to become empty
        while(!(UCSR1A & (1 << UDRE1)));
        // Place new data into buffer
        UDR1 = wcData;
        success = 1;
        return(success);
}
```

This code writes string until you add there ‘*’. Example UA_ws(“send data*”). ends string to *

```
char UA_ws(char* txData)
{
        char success = 0;
        int i = 0;

        // While not end of transmit string
        while(txData[i] != '*')
        {
                // Print string one char at a time
                UA_wc(txData[i]);
                // Increment char position in string
                i++;
        }

        success = 1;
        return(success);
}
```

This code read formatted data from Serial buffer UDR. If serial data with format “$ data*” is receifed, code will store data to buffer. Dollar sign, space and star at the end will be discarded. Dollar sign means that formatted data incoming, then comes space which, then data and last end character *. If data is receifed in different format, receifed data is discarded.

```
void UA_rs()
{
  int temp=0;
  // Reset buffer array
  for(int i = 0; i < 100;i++)
  {
    buffer[i] = ' ';
  }
  while (1)
  {
    buffer[temp] = USART_vReceiveByte();
    // if the character received from uart equals to '$'
    // program gets ready to store the data if character after '$' is ' '
    // (space character)      until it receives end characer '*'
    if (buffer[temp] == '$')
    {
      buffer[temp] = USART_vReceiveByte();              // Over-write on '$'
                if (buffer[temp] == ' ')
                {
                   buffer[temp] = USART_vReceiveByte(); // Over-wrie on ' '
                  temp++;
                  while(1)
                  {
                    buffer[temp]=USART_vReceiveByte();
```

```
                if (buffer[temp] == '*') // if character received is '*' stop
                                         // reading
                {
                      sscanf(buffer,"%s",Datab);       // scan and copy the data receive buffer
break;                                 // to Datab array
                }
            temp++;
                if (temp>=99)            // if temp variable gets higher than 99 stop reading
                {                        // this prevents system corrupting and overflaoding
                  break;  // Data receive buffer
                }
               }
              break;
     }
    break;
    }
  break;
  }
}
```

2.5 Compare two strings

String compare function. Strc compares two strings and verifies that they are equal. This code brakes the strings into characters and searches trhough str2 string until first character equal to ref strings first character is found. Code then verifies that next characters are equal to reference string characters. The size value tells the code how many characters have to match before recognizing is as verified string.

```
char strc(char str2[100], char ref[], int size) // Compares receifed
string to reference string
 {
   char correct = 0;
   int x = 0;
   // start loop
   for(int i = 0; i <= 100; i++)
   {
      if(str2[i] != ref[x])       // if str2 aint same as ref set x=0
      {
         x = 0;
      }
      else               // if str2 equalst to ref increment x
      {
         x++;
         if (x >= size)
         {
            correct = 1;
            break;
         }
       }
    }
    return(correct);
}
```

2.6 Convert string to integer

StringToInt function converts value stored as string or array format into integer value. This function requires math.h to be included. Example: int StringValue = StringToInt(”252, 5”); this will return 252 as StringValue.

```
int StringToInt(char* string_, int count)
{
   int Storage[20];
   long Real = 0;
   int n3 = 0;

   for (int n = 0; n <= count; n++)
   {
      if (string_[n] == '*')
      {
         break;
      }
      // Passes only numbers
      if (string_[n] == '0' || string_[n] == '1' || string_[n] == '2' || string_[n]
      == '3'                                  || string_[n] == '4' || string_[n] ==
      '5' || string_[n] == '6' || string_[n] ==              '7'  || string_[n] ==
      '8' || string_[n] == '9')
      {
         Storage[n3] = (string_[n] - '0');
         n3++;
      }
   }
   int Multiplier = 0;
   for (int n2 = 0; n2 <= (n3 - 1); n2++)
   {
      if (Storage[(n3 - 1) - n2] >= 0 && Storage[(n3 - 1) - n2] <= 9)
      {
         if (Storage[(n3 - 1) - n2] == 0)
         {
            Multiplier++;
         }
```

```
            else
            {
               Real = Real + (Storage[(n3 - 1) - n2] * pow(10, Multiplier));
               Multiplier++;
            }
         }
      }
      return Real;
}
```

2.7 Connecting Teensy to Xbee

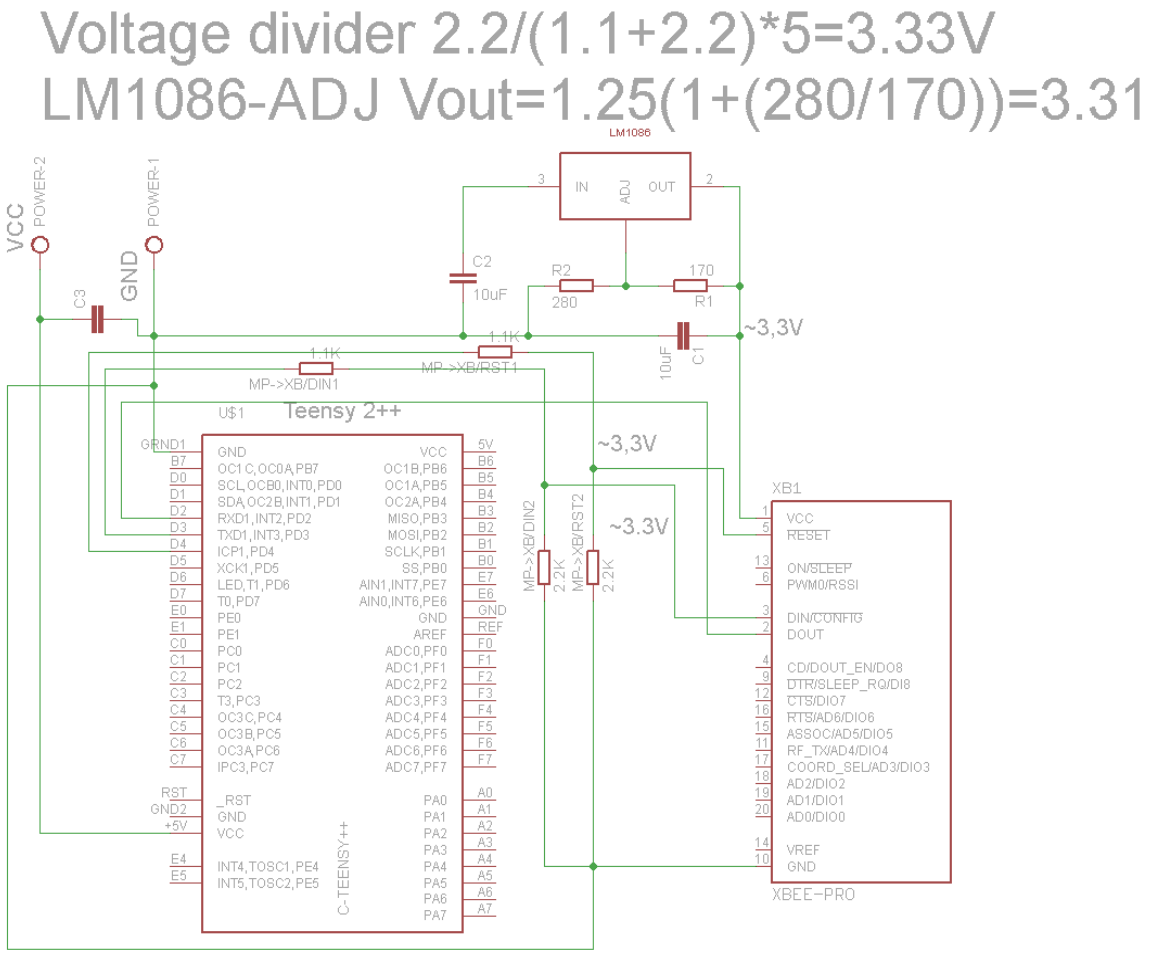
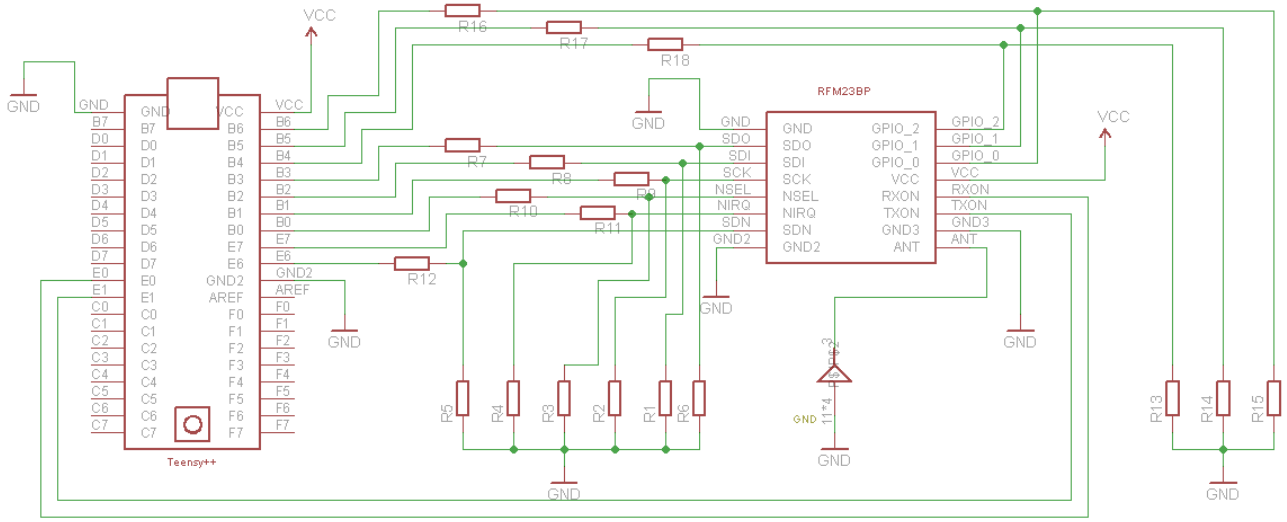


Illustration 5: Teensy<->Xbee

2.8 Hope RFM23BP (Still in progress)

Hope RFM23BP is ism transeiver of multiple frequencies. User can set the frequency by programming into RDM23BP registers. Frequencies to pick 433/868/915MHz with max transmit power +30dBm. Transmit power is also changable. In this tutorial we set the frequency for 434MHz @ 2.4kbytes @13dBm and 868MHz @2.4kbytes @ 27dBm.

RDM23BP works on 5 volts but GPIO works on 3.3 voltage level, thats why in case of teensy we need to make voltage divider for each output of teensy. RDM23BP is controlled through SPI port.



Pin setup

Teensy++	RFM23BP	
MOSI(PB2)	SDI	
MISO(PB3)	SDO	
SCK(PB1)	SCK	
PortB0(PB0)		NSEL
PortE5(PE5)	SDN	
PortE1(PE1)	TXON	
PortE0(PE0)	RXON	
PortE7(PE7)	NIRQ	
PortB6(PB6)		GPIO0
PortB5(PB5)		GPIO1
PortB4(PB4)		GPIO2

//AT90USB646
Code example is taken from [13].
Datasheet can be found from source[14].

```
#include <avr/io.h>
#include <stdint.h>
```

```
#include <avr/delay.h>
#include "RFM23BP.h"
 // We create a class for RFM23BP later on
// Lets define the ports into easier to remember format.

//SPI
#define MOSI PB2
#define MISO PB3
#define SCK PB1
//Chip select
#define NSEL PB0
#define SDN PE5
#define TXON PE1
#define RXON PE0
#define NIRQ PE7
//GPIO
#define GPIO0 PB6
#define GPIO1 PB5
#define GPIO2 PB4


int main(void) // We set the main function
{
// Next we initialize port input and ouputs
    DDRB = (1 << MOSI) | (1 << SCK) | (1 << NSEL) | (1 << GPIO0) | (1 << GPIO1) |
   (1 << GPIO2); // As output
   DDRB &= ~(1 << MISO);          // As input
   DDRE |= (1 << NSEL); // As output
   DDRE = ~(1 << TXON) | ~(1 << RXON) | ~(1 << NIRQ); // As input
    RF.Init(); // This function initializes rfm23bp module
   RF.SetTXPower(27); // 500mW
    while(1)
    {
        RF.Send("Hello", 5);
                       _delay_ms(2000);
```

```
    }
}
// Right Click RFM23Module project → Add → Add New Class → name it as RFM23BP



char RFM23BP::Init(int Power, int Frequency) {
PORTB |= (1 << SDN);
_delay_ms(100);
PORTB &= ~(1 << SDN);
_delay_ms(1000);
Sl.SPIInitialize(2,1); // Atmega8, Master
Reset();
SPIWrite(InterruptEnable2, 0x00); // Clear interrupts
SPIWrite(OMFC, 0x01 );  // disable lbd, wakeup timer, use internal 32769, xton = 1,
into ready mode
SPIWrite(CrystalOscillatorLoadCapacitance, 0x7f); // c = 12.5p
SPIWrite(MCUClockOutput, 0x05); // 2MHz

SPIWrite(GPIO0Confg, 0xf4);    // GPIO0 is for RX data output
SPIWrite(GPIO1Confg, 0xef);    // GPIO1 is TX/RX data CLK output
SPIWrite(GPIO2Confg, 0x00);    // GPIO2 for MCLK output
SPIWrite(IOPortConfg, 0x00);    // GPIO port use default value

SPIWrite(ADCConfg, 0x70);   // No ADC used
SPIWrite(ADCSensAmpOffs, 0x00); // Default

SPIWrite(ModulationModeControl1, 0x20);  // Disable manchest

SPIWrite(TempCal, 0x00);   // Temp off
SPIWrite(TempOff, 0x00);    // No temperature sensor on
SPIWrite(AFCLoopGearshiftOverride, 0x00); // enable afc
SPIWrite(IFFilterBandwidth, 0x1d); // Data Rate 2.4kbytes

if(Power == 30) {
            SPIWrite(TXPower, 0x07); // Max Power
```

```
} else if(Power == 29) {
            SPIWrite(TXPower, 0x06);
} else if(Power == 28) {
            SPIWrite(TXPower, 0x05);
}
   SPIWrite(ClockRecoveryOversamplingRatio, 0xA1); // Calculate from datasheet
   SPIWrite(ClockRecoveryOffset2, 0x20);   //rxosr[10--8]; stalltr = default,
ccoff(19:16) = 0;
   SPIWrite(ClockRecoveryOffset1, 0x4e);   // ncoff = 5033 = 0x13a9
   SPIWrite(ClockRecoveryOffset0, 0xa5);   //
   SPIWrite(ClockRecoveryTimingLoopGain1, 0x00);
   SPIWrite(ClockRecoveryTimingLoopGain0, 0x00);

   SPIWrite(OOKCounterValue1, 0x00);
   SPIWrite(OOKCounterValue2, 0x00);
   SPIWrite(SlicerPeakHold, 0x00);
   SPIWrite(AFCLimiter, 0x1e);

   SPIWrite(DataAccessControl,0x8c );  // Enable packet handler, msb first, enable
crc
   SPIWrite(HeaderControl1, 0x88); // Enable for header byte 0, 1, 2, 3 , receive
   // header check for byte 0, 1, 2, 3
   SPIWrite(HeaderControl2, 0x42); // Header 3, 2, 1, 0 used for head length, fixed
packet length
   // synchronzie word length 3,2
   SPIWrite(PreambleLength, 64);   // 64 nibble = 32byte preamble
   SPIWrite(PreambleDetectionControl, 0x20);   // need to detect 20bit preamble

   SPIWrite(SyncWord0, 0x00);  // Synchronize word
   SPIWrite(SyncWord1, 0x00);
   SPIWrite(SyncWord2, 0xD4);
   SPIWrite(SyncWord3, 0x2D);

   SPIWrite(CheckHeader3, 's');
   SPIWrite(CheckHeader2, 'o');
   SPIWrite(CheckHeader1, 'n');
```

```
   SPIWrite(CheckHeader0, 'g');


   SPIWrite(TransmitPacketLength, 17);
   SPIWrite(TransmitHeader3, 's');
   SPIWrite(TransmitHeader2, 'o');
   SPIWrite(TransmitHeader1, 'n');
   SPIWrite(TransmitHeader0, 'g');


   SPIWrite(HeaderEnable3, 0xff);  // All the bit to be checked
   SPIWrite(HeaderEnable2, 0xff);  // All the bit to be checked
   SPIWrite(HeaderEnable1, 0xff);  // All the bit to be checked
   SPIWrite(HeaderEnable0, 0xff);  // All the bit to be checked


   SPIWrite(TXDataRate1, 0x13); // 2.4kbps
   SPIWrite(TXDataRate0, 0xA9);
   SPIWrite(FrequencyHoppingChannelSelect, 0x00);  // No hopping
   SPIWrite(FrequencyHoppingStepSize, 0x00);   // No hopping


   SPIWrite(FrequencyDeviation, 0x20);
   SPIWrite(ModulationModeControl2, 0x63);  // GFSK, fd[8] = 0, no invert for Tx/Rx
data, fifo mode, txclk -->gpio
   SPIWrite(FrequencyOffset1, 0x00);
   SPIWrite(FrequencyOffset2, 0x00);
   /*

https://www.viestintavirasto.fi/taajuudet/radiotaajuuksienkaytto/taajuusjakotaulukko
.html
    ISM:    434MHz  -> 433.050 - 434.790 MHz (1.740MHz) 25mW ERP  10%
    ISM:    868MHz  -> 869.400 - 869.650 MHz (0.250MHz) 500mW ERP 10%
    ISM:    2.4GHz  -> 2400.00 - 2483.500 MHz (83.500MHz) 10mW ERP
    */


    if(Frequency == 868) {
        SPIWrite(FrequencyBandSelect, 0x6b);
        SPIWrite(NominalCarrierFrequency1, 0x6a );
        SPIWrite(NominalCarrierFrequency, 0x90);
```

```
    } else if(Frequency == 434) {
        SPIWrite(FrequencyBandSelect, 0x53);
        SPIWrite(NominalCarrierFrequency1, 0x64);
        SPIWrite(NominalCarrierFrequency, 0x00);
    }
   SPIWrite(OMFC, ReadyMode);
        /* validate Device Type and confirm Device version */
    if(SPIRead(DeviceType) == 8 && SPIRead(DeviceVers) == 6) {
        return 1;
    }
    else
    {
        return 0;
    }

   return 1;
}
```

3 Atmel AVR

3.1 Timers

We start by configuring timer 0 of Atmega 2560. Timer 0 is 8 bit timer thus contains 2^8 bytes = 255 states. TCTN0 is timer 0 memory. When timer ticks from 0 → 255 overflow event happens. OverFlow Flag(TOV) will be set high.

You can set value to TCNT0 to start from desired count. For example, setting TCNT0 to 125 means that from 125 → 255 causes OverFlow. It's also possible to set value to OCR0(output compare) register. If we set OCR0 to 200 when TCNT0 125 count reaches to 200, OCF0 (Output Compare) Flag is set high.

All register are are shown in datasheet of Atmega 2560 → http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf

3.2 Configuring timers

Good reference to timers: https://exploreembedded.com/wiki/AVR_Timer_programming

Configuring timer is done by editing TCCR0.

Frequency of clock source using prescalers CS02, CS01,CS00 and setting mode of timer with WGM00 & WGM01. Check datasheet for TCCR0 register.

Prescaler choices:

CS02	CS01	CS00	
0	0	0	No clock(Stopped)
0	0	1	CLK
0	1	0	CLK/8
0	1	1	CLK/64
1	0	0	CLK/256
1	0	1	CLK/1024
1	1	0	CLK/TO-Falling edge
1	1	1	CLK/TO-Rising edge
WGM00	WGM01	Mode	
0	0	Normal	
0	1	CTC(Clear timer on compare match)	
1	0	PWM(Phase correct)	
1	1	Fast PWM	

TIFR register contain interrupt flags.

3.2.1 One second timer 0 example with 8 bit timer

Atmega 2560 runs on 16MHz. Lets calculate how long does it take to cause overflow.

Timer = CPU Frequency / Prescaler

Timer = 16MHz/1024 = 15,625KHz → 1/15,625KHz = 64uS

→ 64uS*255 = 16mS → It takes 16mS to cause overflow

Timer = 16MHz/8 = 2MHz → 1/2MHz = 0.5uS

→ 0.5uS * 255 = 127uS → Takes 127uS to cause overflow

We want to create one second timer so one second is x amount of overflows.

X = Desired time / OverFlow time -> 1s/16ms = 62.5 overflows.

X = Desired time / OverFlow time → 1s/127uS = 7874.016 overflows.

In this case choosing prescaler of 8 would be more accurate.

Create new arduino sketch. This code will print to terminal ” One second timer” every second.

```
Int timer_over_flow_count = 0;

Void setup() {
        Serial.begin(9600);
        noInterrupts(); // Disable interrupts
        TCNT0 = 0x00;
        TCCR0 = (1 << CS00)| ( 1 << CS02);          // Prescaler 256
        interrupts(); // Enable interrupts
}

void loop() {
        while((TIFR & 0x01) == 0); // Wait for overflow flag to go high
        TCNT0 = 0x00;
        TIFR = 0x01; // Clear timer 1 overflow flag
        timer_over_flow_count++;
        if( timer_over_flow_count >= 62) {
                Serial.println(” One second timer ”);
                timer_over_flow_count = 0;
        }
}
```

3.2.2 One second timer 1 example with 16 bit timer

Difference to timer 0 is that we have 16 bits instead of 8 thus we have 2^16 states

2^16 → 65536

Atmega 2560 runs on 16MHz. Lets calculate how long does it take to cause overflow.

Timer = CPU Frequency / Prescaler

Timer = 16MHz/1024 = 15,625KHz → 1/15,625KHz = 64uS

→ 64uS*65536 = 4194 mS → It takes 4.2s to cause overflow

We want to create one second timer so one second is x amount of overflows.

X = Desired time / OverFlow time -> 1s/4.2s = 0.24 overflows.

Load OCR1A register, and monitor OCF flag.

Set prescaler to 1024 with CS12 and CS10.

Lets calculate how many ticks are needed to get 1 second.

1s/64uS = 15625

```
void setup() {
        Serial.begin(9600)
        noInterrupts();                                 // Disable interrupts
        TCCR1B = (1 << CS10) | (1 << CS12)              // Prescaler 1024
        OCR1A = 15625;                                  // 1second delay
        TCNT1 = 0;
}

void loop() {
        while()TIFR & (1 << OCF1A)) == 0);              // Wait for overflow flag
        Serial.println(”1 second”);
        TCNT1 = 0;
        TIFR |= (1 << OCF1A);               // Clear timer 1 overflow flag
}
```

3.2.3 1mS Overflow Interrupt timer with timer 1

Here we create 1mS interrupt timer with overflow. Timer 1 is 16 bit timer thus capacity for the timer 2^16 = 65536. Lets calculate what needs to bea laoded to TCNT1. When timer is ticking it keeps increasing TCNT1 register with 1 on each cycle. Thus we need load a value 65536 – desired time in ticks.

Timer 1 = 16bit → 2^16

Timer 1 = 65536

Time to overflow = 65536 * (1/(Timer frequency/Prescaler value))

=Timer frequency / Prescaler

= 16MHz / 1024 = 15.625kHz = 1/15.625kHz = 64uS

Time for 1 tick period 64uS.

Time to overflow = 65536 * 64uS = 4.194S = 4.2S to cause overflow.

Lets calculate how many ticks are needed for 1mS

1mS/64uS = 15.625 ticks. This is not most accurate due to every 1ms 0.385 drift is caused. You can minimze this by adjusting the prescale value.

As we calculcated we need to load value of 65536 – 15.625 to TCNT1

Include necessary files.

```
#include<avr/io.h>
#include<avr/interrupt.h>
```

```
ISR(TIMER1_OVF_vect)  {                 // Interrupt routine
                TCNT1 = 65520 // Reload timer
}

void setup() {
                TCCR1A = 0x00;
                TCCR1B = (1 << CS10) | (1 << CS12); // Timer mode with 1024 prescaler
                TIMSK = (1 << TOIE1); // Enable timer1 overflow interrupt (TOIE1)
                TCNT1 = 65520;
                sei(); // Enable global interrupts
}
```

Good ref https://exploreembedded.com/wiki/AVR_Timer_InterruptsInterrupt timer with timer 1 #TODO

4 Arduino

As notice the pin definition depends on your board. If not seperatly specified I’m using STM32 board as Arduino. The arduino board may differ from what you have but basic concept it stil lthe same just different pins. Also always check the output votlage of pins most chips are 3.3V not 5V as Original Arduino is.

For example Arduino Mega pinouts are 5V tolerant and will output voltage of 5V. STM32, MAPLE Mini board is 3.3V tolerant and will output voltage of 3.3V. If pin is 5V tolerant it only means it can endure 5 volts this doesn’t say what does it output. Always check this from datasheet.

4.1 Choosing development board

Arduino used in the examples is either Arduino Mega or Maple Mini(STM32). I personally prefer to use Maple Mini because its compact size and usability. Price for Maple Mini is about 2.4 – 3 € price for genuine Arduino Mega 35€. Maple Mini is a bit more complicated to start with than genuine board but once environment theres no difference to using. It actually becomes easier to work on Maple due to 3.3V pin outputs. Maple processing power is 72MHz and Arduino Mega 16MHz so I assume ya understand why I prefer Maple.

To buy Maple Mini you can buy it from Aliexpress. It takes quite a while to arrive but in case you need more than one arduino, you save a lot of money.

I explain here how to install Arduino environment for Ubuntu and Ubuntu supporting Maple Mini.

Start buy downloading Arduino IDE. Download and install.

Download STM32 arduino suppport from stm32duino.

Open your Arduino folder.

Open hardware folder and extract/unzip downloaded STM32duino files here.

– If Ubuntu--

Open the Arduino_STM32-master or your extracted folder.

Under folder tools you can see folder linux.

Runt the install.sh file in linux folder

$./install.sh

You might have to run this in sudo but try first without.

Once installation is done open Árduino IDE.

--if Windows--

#TODO

From Arduino IDE select Tools→Board→Board Manager and install the Arduino SAM Boards(32-bits ARM Cortex-M3).

Once installed restart Arduino IDE. From Tools→Board select Maple Mini.

Plug the cable to Maple and press first reset and then but button. Blue led should start flashing on even phase. Now you have driven Maple to programming state and can upload your arduino software.

4.2 Blink led

For this section you only need a Maple mini board. Maple mini contains a led at pin number 33. We can use this to make sure development board works.

```
#define LED_PIN 33
void setup() {
		pinMode(LED_PIN, OUTPUT);
}

void loop() {
		digitalWrite(LED_PIN, LOW);
		delay(500); // Wait for 500ms
		digitalWrite(LED_PIN, HIGH);
		delay(500); // Wait for 500ms
}
```

Press the → button to compile and run. Now you should see LED blinking every 500ms

4.3 Async led blink with counter

In this section we blink led without hindering rest of the program. On previous example we use delay. This causes whole program to get stuck to delay for defined time. As in previous example delay is 500ms, so for 500ms our program does nothing. This is not very practical. We can use variable to count how many loops has went trough and we minimize the delay to 1ms which is acceptable in this example. It’s always best to not uses delays at all but that we go trough later.

```
#define LED_PIN 33
int counter = 0;

void setup() {
		pinMode(9600);
}

void loop() {
		delay(1); // We delay microcontroller with 1ms each loop.
```

```
        counter++; // Increase counter by 1 on each loop
        if(counter >= 1000) {
                if(digitalRead(LED_PIN) == LOW) {
                        digitalWrite(LED_PIN, HIGH)
                } else {
                        digitalWrite(LED_PIN, LOW)
                }
                counter = 0;
        }
}
```

4.4 USB Serial hello world

USB communication is crucial for debugging. When building large programs usb communication is used to debug when problems rises. If a program tends to get stuck somewhere or you are not getting correct data, usb can be used to print where it is getting stuck or what data is in the buffer.

Lets use the previous timer and make the program print ”hello” every second.

```
int counter = 0;
void setup() {
        Serial.begin(9600); // Set baudrate to 9600
}

void loop(){
        delay(1);
        counter++;
        if(counter > 999) {
                Serial.println(”Hello world”);
                counter = 0;
        }
}
```

4.5 USB Serial echo

This program prints back everything you send with arduino terminal.

```
void setup() {
		Serial.begin(9600);
}

void loop() {
		if(Serial.available() > 0) {
			Serial.println((char)Serial.read());
		}
}
```

(char) tells programmer to handle Serial.read() values as char. If you send a ASCII ’a’ this will be printed in decimals. With (char) this will be printed back to terminal as ’a’. This is called casting. More info of casting from tutorialspoint.

4.6 Timing systems

In this section we build simple timing system with original Arduino Mega. Arduino Mega digital input and outputs are 5V tolerant and will output 5V. Our device will start timing once runner passes start and stop once runner passes goal.

As display we use PC and as gates we use buttons.

Following are required for this example

- 2x 10k to 10M Resistors
- Arduino
- USB cable compatible with arduino
- Wire
- Brakeout board
- 2xPush buttons

To draw the circuit I’m using Circuit Diagram

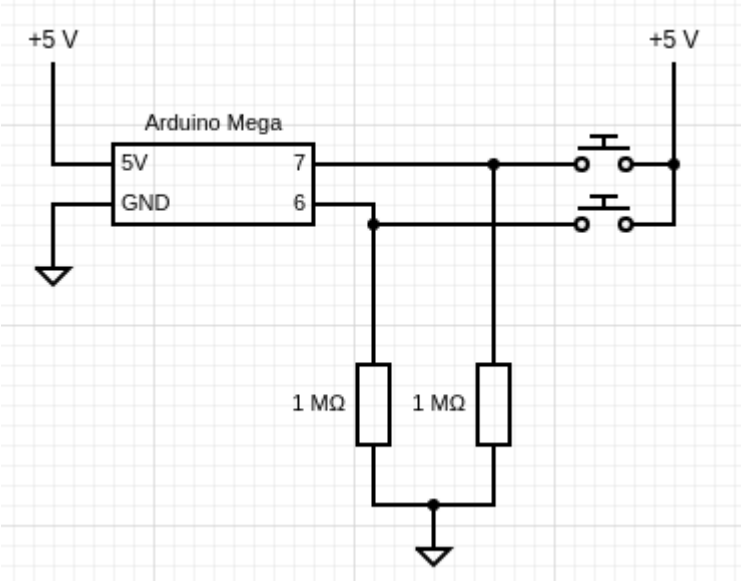


Illustration 6: Circuit design: Timing System Simple

Let’s make a circuit as in Illustration 7. Arduino pin 7 will be start and pin 6 goal. When Push button connected to pin 7 is pushed timer will start and button on pin 6 stop. Resistors are needed to pull DIO(Arduino digital input output pins) low. If you don’t add the resistor DIO will be floating when button is not pressed. This could lead arduino not recognizing if voltage is low and migh think button is activated constantly. When we pull DIO low with resistor it will ensure that voltage level is low and pin is grounded properly.

In the circuit when we press start button(button connected to pin 7) or goal button, current will flow to DIO pin and it will recognize high voltage(+5V). We can read these votlage changes with digitalRead(pin) command. Make design on brakeout board and lets start coding.

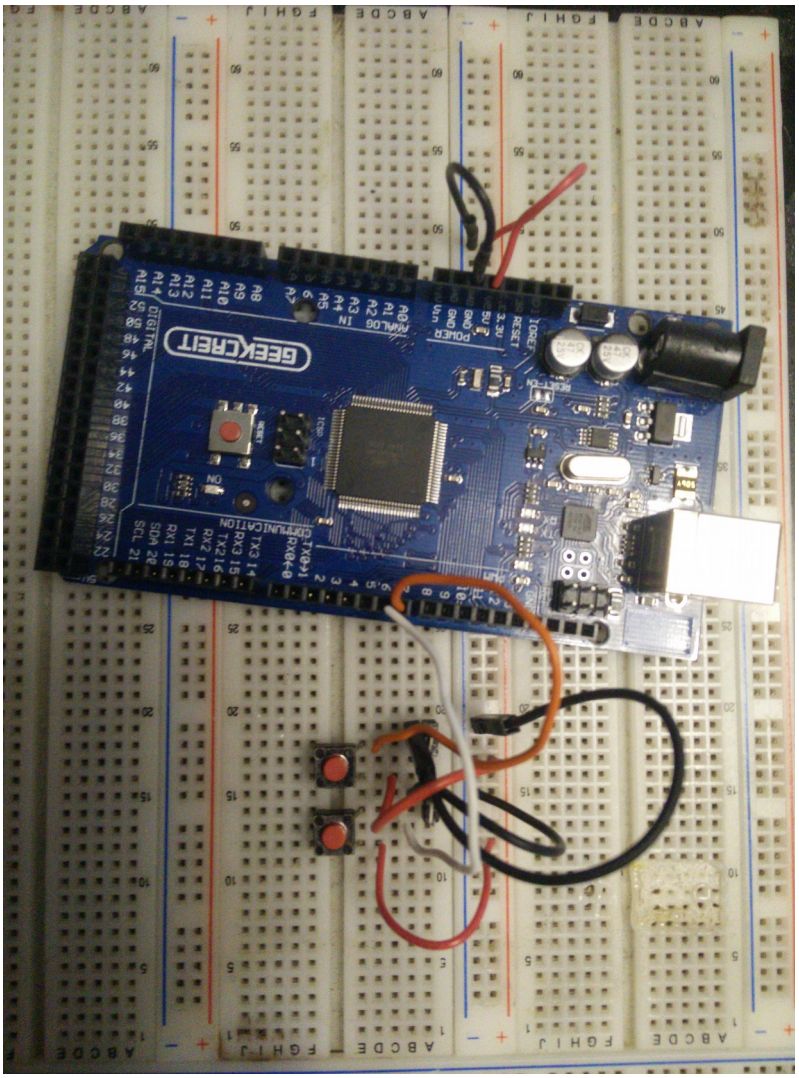


Illustration 7: TS on brakeout

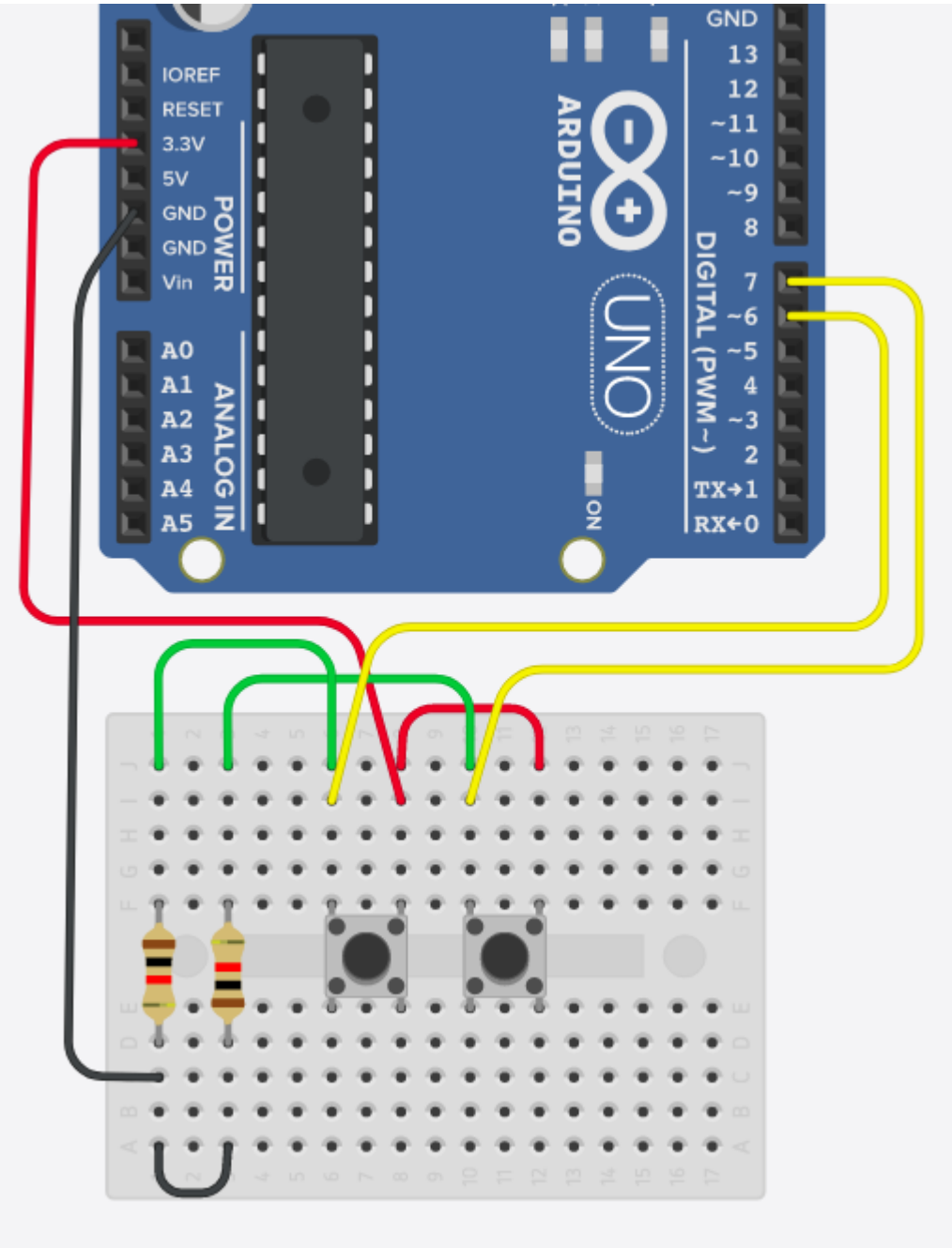


Illustration 8: Timing system simple 3D

Start a new sketch in Arduino IDE from File→New. Select correct Board Tools→Board→Arduino Genuine Mega or Mega 2560.

We also need to add counter for timer. Define variable timeCounter as long. Integer does not have enough memory to containt timed data. In our case integer could hold until 65 seconds. Integer is 16bits thus 65536 bits. We want to store every 1mS 65536/1000mS →65.536 seconds. Long is 32bits → 2^32/1000 and can contain time data until 1193 hours.

We start by defining our pins.

```
#define START 7
#define STOP 6Timing systems
long timeCounter = 0;                   // Keeps track on current time
bool timeRunning = false;               // Tells us if timer is running
```

Initialize pheripherals. We need exact and as accurate timer as we can thus we are using hardaware timers(more detailed in AVR section). In this one we set 1mS timer with overflow timer. To calculate → 1/(16MHz / 64(prescaler)) = 4uS → 1mS/4uS = 250ticks. ATMEGA2560 datasheet. We need to load to TCNT1 65536 - 250

```
Void setup() {
                // put your setup code here,Timing systems  to run once:
                Serial.begin(9600);
                pinMode(START, INPUT);
                pinMode(STOP, INPUT);

                noInterrupts();
                // Clear registers
                TCCR1A = 0;

                TCNT1 = 65286;  // Preload timer
                TCCR1B |= (1 << CS10) | (1 << CS11); // 64 Prescaler
                TIMSK1 |= (1 << TOIE1); // Enable timer 1 overflow timer
                interrupts(); // Enable interrupts.
}

ISR(TIMER1_OVF_vect)  // Interrupt service routine, called when overflow happens
{
                TCNT1 = 65286;          // Reset overflow timer
                timeCounter++;
}

void loop() {#define START 7
#define STOP 6Timing systems
long timeCounter = 0;                   // Keeps track on current time
bool timeRunning = false;               // Tells us if timer is running
```

```
        if(digitalRead(START) && !timeRunning) {
                timeCounter = 0;
                timeRunning = true;
                Serial.println("Timer started");
        }

        if(digitalRead(STTiming systems OP) && timeRunning) {
                timeRunning = 0;
                Serial.print("Counted time: ");
                Serial.println(timeCounter, DEC);
                timeCounter = 0;
        }
}
```

Combile the software to Arduino and run. Open serial monitor. When you push start Time stated will be displayed in serial monitor. When you push stop timer stops and counted time will be displayed in milliseconds.

4.6.1 Timing systems with constant time display

We use the previous example but we add a constant time display. We want to display the running time when someone is taking time. To make this we need add cases when to display for the program.

Lets make the logic as follows. When timer starts and timeRunning goes high we start displaying the time for user. We don’t wanna display it every milliseconds so we make it display every 100mS. For this we need to make another variable to keep count of milliseconds and every 100mS trigger display event.

Add variable display timer under timeRunning variable. Your defines should look something like this. Let’s define it as char to save some memory. Char variable only eats 2^8 bytes.

```
#define START 7
#define STOP 6
long timeCounter = 0;
bool timeRunning = false;
char displayCounter = 0;
```

Everytime 1mS is gone we want to store this to displayCounter but only if someones time is being tracked. Thus we can use timeRunning as indicator when to cound display time.

```
ISR(TIMER1_OVF_vect) // Interrupt service routine
{
	TCNT1 = 65286;
	timeCounter++;
	if(timeRunning) displayCounter++;
}
```

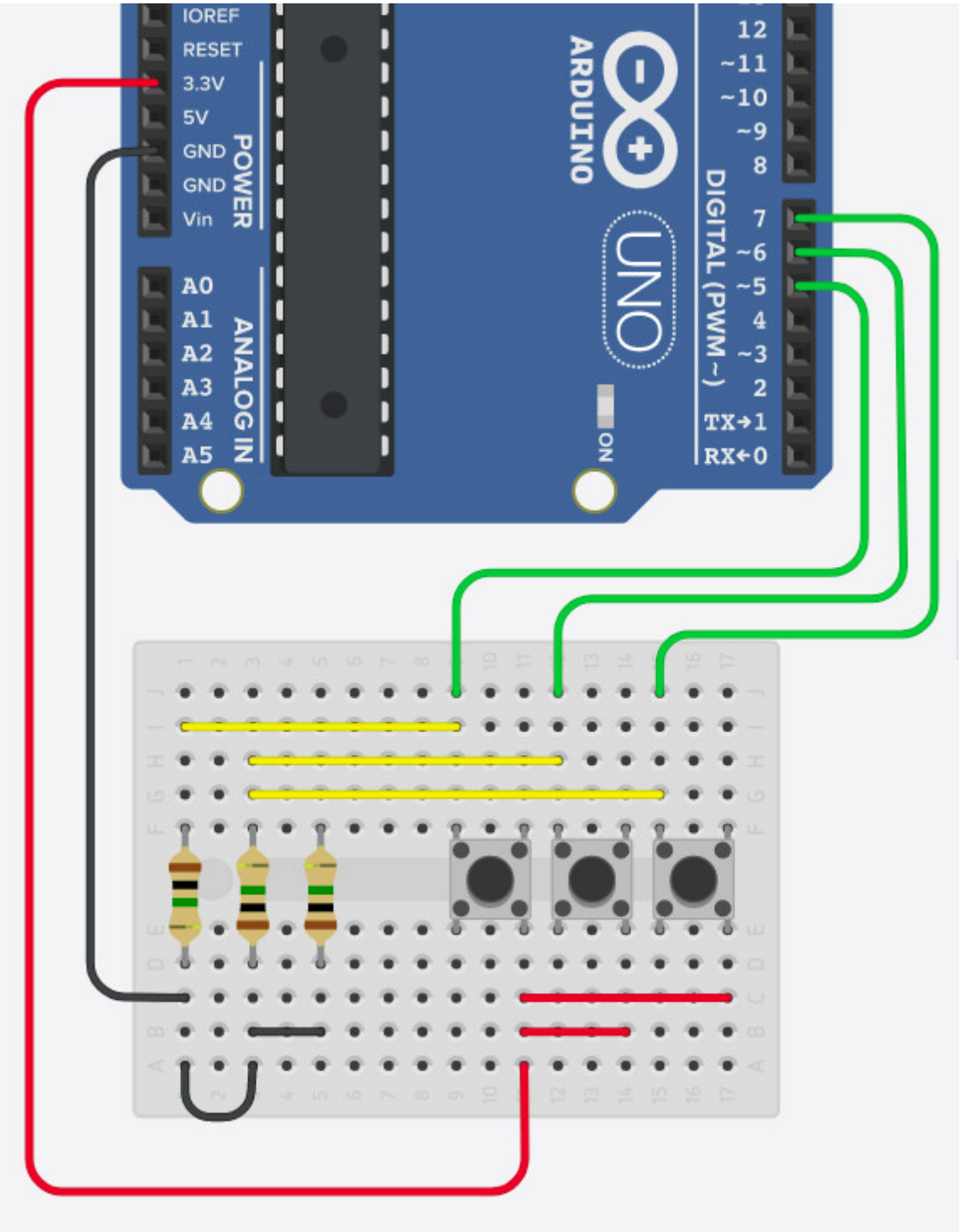
We need to add to main program event when to display running time. For this lets give condition. If time is running and displayTimer has reached 100ms, print time to pc and reset display timer. Our loop program should look like this.

```
void loop() {
	if(digitalRead(START) && !timeRunning) {
		timeCounter = 0;
		timeRunning = true;
		displayCounter = 0;
		Serial.println("Timer started");
	}

	if(timeRunning && (displayCounter >= 100)){
		Serial.print("Running time is: ");
		Serial.println(timeCounter, DEC);
		displayCounter = 0;
	}

	if(digitalRead(STOP) && timeRunning) {
		timeRunning = 0;
		Serial.print("Counted time: ");
		Serial.println(timeCounter, DEC);
		timeCounter = 0;
	}
```

```
}
```

When you run this software on Arduino running time should be displayed.

4.6.2 Split timer



4.7 Distance meter used for measuring jump heights

Requirements:

- Arduino
- HC-SR04
- HM-10

In this project we are using Arduino Mega 2560, HM-10 Bluetooth low energy and HC-SR04 ultrasound sensor.

HC-SR04 sends a ultrasound and waits for reflection from anysurface. Once reflection of sound is detected we can use the duration between sending and receiving to calculate travel distance.

This project can be done under 15€. You can buy from aliexpress Arduino compatible MAPLE mini. STM32 based dev board and order HM-10, HC-SR04 from aliexpress.

Receiving units will take from 2weeks to 4 months but you save a lot of money.

How to setup STM32 based MAPLE mini will be added later to the tutorial.

Ideology behind height meter is as follows:

- Calibrate system, find highest point or infinity point
- Detect if any activity happens
- Find lowest and highest distance during activity
- Display this at android phone

Start a new sketch, lets call it Distance_measure.

Connect HM-10 to UART2.

Connect HC-SR04 trigger to pin 12 and echo to pin 11.

Connect 5v and gnd to both modules.

Define trigger and echo point. Trigger pin send ultrasound from HC-SR04 and echo pin reads the reflection.

```
const int trigPin = 12;
const int echoPin = 11;
```

Defines max amount of samples taken

```
const int sampleTableSize = 1000;
```

Defines reuqired amount of samples for calibration

```
const int calibrateSamples = 500;
```

Keeps track of calibration state.

```
boolean calibrateSuccess = false;
```

Define max time to wait for sample/reflection 2 seconds

```
const int sampleTime = 2;
```

Table to hold samples in memory.

```
int sampleTable[sampleTableSize];
```

Container to keep track how many samples were taken.

```
int sampleTableCount = 0;
```

timer keeps count of passed time

```
int timer = 0;
```

Controls if timer is enabled or not

```
boolean timerEnabled = false;
```

calibratedValue is set by calibration. This represents most common values ultrasound sensor is reading. It’s necesary to find out default point, what we use to detect activity.

```
Int infinityValue = 0;
```

Printed results tells us how many results have we had.

```
int printedResults = 0;
```

Outdoor boolean is set true if while calibration infinity is read. Meaning no reflection was found. We can assume that either the roof is higher than ultrasound sensor operating range or we are outside.

```
boolean outdoor = false;
```

Max operating range 2cm – 400cm, added marginal of 20.

```
const int maxOperatingRange = 420;
```

Set area how much sample can difDistance meter used for measuring jump heights #Under progressfer before recognized as acitivity. Now it’s set as +15 -- -15 from infinityValue.

```
const int detectionRate = 30;
```

```
void setup() {
		pinMode(trigPin, OUTPUT); // define trigPin as output
		pinMode(echoPin, INPUT); // define echoPin as input

		Serial2.begin(9600); // Init baudrate for bluetooth
		// Configure timer 1 with interrupt, more info at timers chapter
		noInterrupts();
		TCCR1A = 0;
		TCCR1B = 0;
		TCNT1 = 0;

		OCR1A = 31250;
		TCCR1B |= (1 << WGM12);   // CTC mode
		TCCR1B |= (1 << CS12);    // 256 prescaler
		TIMSK1 |= (1 << OCIE1A);  // Enable timer compare interrupt
		interrupts();
		delay(5000); // Allow 5s of delay before calibration is started
}
```

// Interrupt

```
ISR(TIMER1_COMPA_vect) {
  if(timerEnabled) {      // If timer enabled read count
    timer++;              // Add 1 every time interrupt is called
  }
}
```

Read distance sends triggers ultrasound sensor and waits for reflection. Returns distance as integer. Equation to count distance ref:

https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

When ultrasound travels from sensor it goes twice the distance we want to know.

Distance to the object and distance coming back to sensor.

v = 340m/s, speed of sound

v = 0.034cm/uS

Time = distance / speed = 10 / 0,034 = 294uS

Total Distance = time * speed = 294uS * 0,034cm/uS = 10cm

Distance to object = Total Distance / 2 = 5cm

```
int readDistance() {
		long duration = 0;
		int distance = 0;
		digitalWrite(trigPin, LOW); // Ensure trigPin is low
		delayMicroseconds(2);
		// Activate ultrasound
		digitalWrite(trigPin, HIGH);
		delayMicroseconds(10); // 10uS required to activate pulse
		digitalWrite(trigPin, LOW);
		duration = pulseIn(echoPin, HIGH); // Wait until echoPin goest high;
		// count distance
		distance = duration * 0.034/2;
		return distance;
}
```

Calibrate function takes multiple samples from ultrasound sensors. Finds value or distance which doesn’t variate and assumes that this is roof. If roof is higher than operating range or we are outside, reflection cannot be read. In this situation we set outdoor variable true and assume on default nothing can be read.

```
void calibrate() {
		Serial2.println("Calibrating"); // Tell us on mobile phone what’s happening
		int lowest = 0;
		int highest = 0;
```

```
		int sample = 0;

		for(int i = 0; i < calibrateSamples; i++)  {
			sample = readDistance();
			if(lowest == 0 || lowest > sample )   {
				lowest = sample;
			} else if(highest == 0 || highest > sample) {
				highest = sample;
			}

			int difference = (highest - lowest);

//If highest and low difference is less than 5 and highest detected point is less than 420cm
			if(difference > 5 && highest < maxOperatingRange) {
				Serial2.println("Failed to calibrate");
				calibrateSuccess = false;
				return;
			}
//If highest point is over the operating range, consider system being outdoor
			if(highest > maxOperatingRange) {
				outdoor = true;
				break;
			}
		}
		infinityValue = sample;
		calibrateSuccess = true;

		//Print to android whats our status and value
		Serial2.print("Calibration success ");
		Serial2.println(infinityValue);
}
```

Detect action function monitors if activity is sensed. Depending on calibration if calibrated with infinityValue or as outdoor mode. If we are monitoring infinityValue then detectionRate determines when activity is detected. If detection rate is set as 30,

and infinity value is 160cm. For example roof from your table is 160cm and detectionRate is 30. If read sample is less than 145 or higher than 175 then we see this as activity.

In case of outdoor mode anything less than maxOperatingRange, 420 in this case is seen as acitivty.

```
boolean detectAction() {
		// Detect if something is happening
		int sample = readDistance();
		int getDifference = ((int)sample - (int)infinityValue)
		if(sample < maxOperatingRange && outdoor) {
			 sampleTable[0] = sample;
			return true;
		} else if(!outdoor && (getDifference > detectionRate/2
			|| getDifference < -detectionRate/2))  {
			sampleTable[0] = sample;
			return true;
		}
		return false;
}
```

Allow timer variable to increase inside interrupt call.

```
void timerStart() {
		timer = 0;
		timerEnabled = true;
}
```

Disable counting time

```
void timerStop() {
		timerEnabled = false;
}
```

Filter distance uses very simple ways to filter active samples. Function needs to be called only if action detected. Indentation is here quite poor due to space :).

If in inside mode that we have detected a calibrateable value. Meaning we have calibrated value inside operating range. We start timer. If we get no reflection or samples inside 2 second timer will shutdown this function and return latest value.

If 6 of the taken samples are near calibratedValue(infinityValue) we consider that object has disappeard from operating area and stop taking samples.

If we are in outdoor mode and 6 of taken samples are higher than max operating range. We consider object has disappeard and stop taking samples.

```
int filterDistance() {
		boolean finished = false;
		timerStart();
		int samplePoll = 1;
		while(timer < (sampleTime * 2)) {

		Take samples until timer has triggered or infinity is detected
		sampleTable[samplePoll] = readDistance();
		samplePoll++;
		if(samplePoll > 5) {
		int count = 0;
		for(int i = 0; i <= 6; i++) {
		if( sampleTable[samplePoll -i] > maxOperatingRange && outdoor) {
				count++;
		} else if( !outdoor && sampleTable[samplePoll -i] - infinityValue
				< detectionRate && sampleTable[samplePoll -i] - infinityValue
				> -detectionRate) {
		count++;
		}
		}
		if(count >= 5 || (samplePoll + 5) > sampleTableSize) {
		sampleTableCount = (samplePoll - 5);
		break;
		} else {
		 sampleTableCount = (samplePoll - 1);
		}
		}
```

```
                }
                timerStop();
}
```

Find lowest point from taken samples and filter out 0 values.

```
int getLowPoint() {
  int lowPoint = 0;
  for(int i = 0; i < sampleTableCount; i++) {
    if( (sampleTable[i] < lowPoint || lowPoint == 0) && (lowPoint < maxOperatingRange)
) {
      if( sampleTable[i] <= 0) {
         continue;
      }

      lowPoint = sampleTable[i];
    }
  }
  return lowPoint;
}
```

Find highest point from samples and filter out infinity or max operating range values

```
int getHighPoint() {
                int highPoint = 0;
                for(int i = 0; i < sampleTableCount; i++) {
                if(sampleTable[i] > highPoint || highPoint == 0) {
                if( (sampleTable[i] < maxOperatingRange) && outdoor) {
                continue;
                } else if(!outdoor && (sampleTable[i] - infinityValue < detectionRate &&
                sampleTable[i] - infinityValue > -detectionRate) ) {
                continue;
                }
                highPoint = sampleTable[i];
                }
                }
                return highPoint;
}
```

Find avarage distance from sampled values. Filter 0 and infinity/max distances from calculations.

```
int getAvg() {
		long sum = 0;
		int count = 0;
		for(int i = 0; i < sampleTableCount; i++) {
		if( (sampleTable[i] < maxOperatingRange) && outdoor) {
		continue;
		} else if(!outdoor && sampleTable[i] - infinityValue < detectionRate &&
		sampleTable[i] - infinityValue > -detectionRate) {
		continue;
		} else if (sampleTable[i] <= 0) {
		continue;
		}
		sum = sum + sampleTable[i];
		count++;
		}
		sum = sum/count;
		return sum;
}
```

Runs necessary functions to get informative data to user and prints all to bluetooth device.

```
void analyzeSamples() {
		printedResults++;
		Serial2.println("");
		Serial2.print("Results : ");
		Serial2.println(printedResults);
		Serial2.print("Find lowest point: ");
		Serial2.println(getLowPoint());
		Serial2.print("Find highest point: ");
		Serial2.println(getHighPoint());
		Serial2.print("Find avarage point: ");
		Serial2.println(getAvg());
		 Serial2.println("");
```

```
}
```

Main function.

```
void loop() {
		int height = 0;
		while(!calibrateSuccess) {
			calibrate();
		}
		// If action detected start taking samples for x amount of time;
		if(detectAction()) {
			filterDistance();  // For x amount of time take samples
			analyzeSamples();
		}
}
```

4.8 Radio Module: RF4432F27 500mW 868MHz

NiceRF RF4432F27 is very affordable 500mW 868MHz radio tranceiver. Same module is capable of transmitting and receiving with price of 6- 9€ you can find this module from Aliexpress. You can set the correct operating frequency by configuring the module from your program. To select correct frequency check your countrys frequency limits. In most contries ISM band which allows 500mW power operates at end of 868MHz frequency. You also need to check if there are any time limitations. In Finland you are allowed to transmit 500mW transmissions 6 minutes per an hour on ISM frequency. ISM is for Industry Science and Medical applications.

Radio module RF4432F27 can output with power of 500mW. This is very high output power and when antenna is added to the equation power can be almost 1Watt.

To ensure it is within legal limits of EU you can drop the power of the unit through program. Notice the limits can vary depending on the country. Register 0x6D least significant bits determines the power output. All registeres are 8 bits long 0000 0000. By default from manufacturer this chip is set at full power. When least significant bits are in 0111 state full power is transmitted. We want to set it to 0110. 0111 equals to 27dBm and 0110 equals to 24dBm. Power steps are 3dBm.

Power drob here is quite radical so to compensate we need 3dBi antenna. This will increase the transmit power to 27dBm.

Equation to calculate transmitted output power is

Output Power(dBm/W/mW) = Radio Transmit power(dBm) – Loss from Cables & Connectors(dB) + Antenna Gain(dBi)

To convert dBm to mW/W and vice versa:

mW to dBm = $10\text{Log}_{10}(\text{mW})$

dBm to mW = $10^{(\text{dBm}/10)}$

4.9 GSM/GPRS with NEOWAY M590

NEOWAY M590 is very cheap GSM/GPRS module. You can get for 2-3€ these modules on brakeout board from aliexpress. It’s capable for SMS and GPRS data connection. This works of EGSM900/DCS1800, GSM850/1900 and Quad-band. For USA GSM850/1900 for europe EGSM900/DCS1800.

You can connect Arduino to the Module with UART which default baudrate is 115200. On this example I’m using Maple Mini(STM32).

4.9.1 Connect pc serial terminal with NEOWAY

In this example we use Arduino as a adapter between computer and NEOWAY M590. We use serial terminal to communicate with Arduino and arduino then transfers this to NEOWAY.

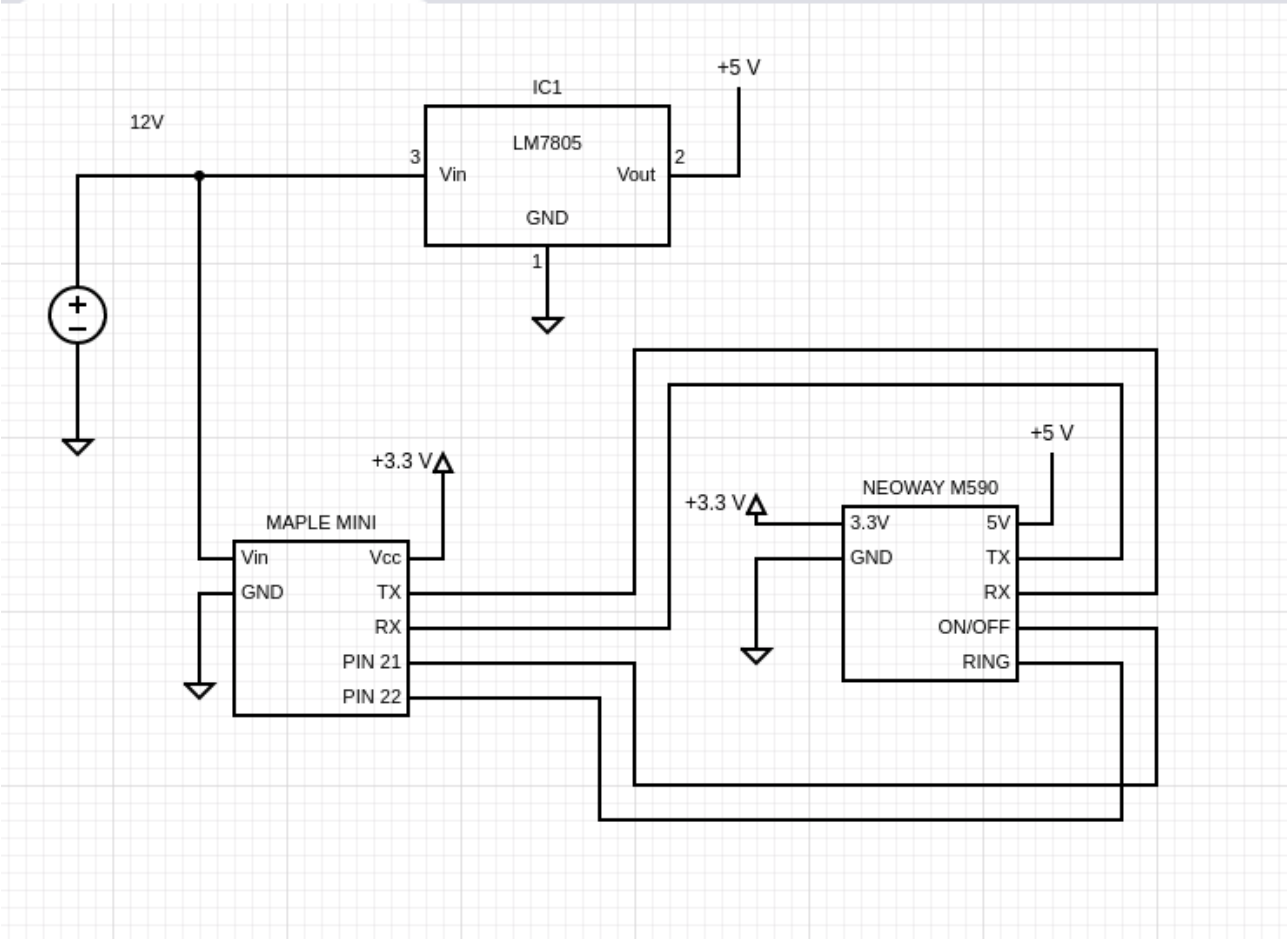


Illustration 9: NEOWAY M590

Start by defining serial interfaces, variables and gsm pins. NEOWAY has a ON/Off and RING pin. We need to pull RING low and ON/OFF high after 300mS from starting. Counter variable increase everytime we receive data from NEOWAY and acts as pointer where to store received character to gsmBuffer. GsmBuffer is our memory for storing incoming data from NEOWAY. DataReceived tells us if data has been received and transferAwait is to give time for reading all data from NEOWAY before printing it to PC.

```
#define pc Serial
#define gsm Serial3
#define gsm_on 21
#define gsm_ring 22
char gsmBuffer[100];
char counter = 0;
bool dataReceived = false;
int transferAwait = 0;
```

Lets initialize our software.

```
void setup() {
		pc.begin(9600); // Start serial with pc
		gsm.begin(115200); // Start serial with NEOWAY default baudrate 115200
		pinMode(gsm_on, OUTPUT); // Set pin mode as output
		pinMode(gsm_ring, OUTPUT); // Set pin mode as output
		initializeGSM(); // Call initializeGSM function
}

void loop() {
		pcToModule(); // call pcToModule function
}
```

This initializes and does start-up sequence for NEOWAY

```
void initializeGSM()
{
		for(int i = 0; i < 100; i++) {
				gsmBuffer[i] = 0x00;
		}

		digitalWrite(gsm_on, LOW);  // Reset ON/OFF pin, pull low
		digitalWrite(gsm_ring, LOW); // Set RING activve
		delay(300); // Wait for 300mS
		digitalWrite(gsm_on, HIGH); // Power on system after 300mS
}
```

PcToModule function handles the way pc and module communicate. Because pc works with baudrate of 9600 and GMS on 115200 I added buffer and transferAwait to ensure proper communication. If we write gsm to pc sameway as pc to gsm this won’t work well due to speed difference. When printing incoming data to PC it takes more time than reading the data from gsm thus missing some packages from GSM.

```
void pcToModule()
{
		// Wait while data is incoming from PC and transfer them to module
		while(pc.available() > 0) {
				gsm.write(pc.read());
		}
```

```
        // Wait while data is coming from gsm and store incoming characters to to
        // buffer
        while(gsm.available() > 0) {
                transferAwait = 0;      // Reset transfer await
                char inChar = ' ';      // Reset and define inChar as char
                inChar = gsm.read();    // Read from gsm
                // Increase counter by 1 everytime storing character to buffer
                gsmBuffer[counter++] = inChar;
                dataReceived = true;    // Set dataReceived true
        }
        // If we have received any data from gsm start counting transferAwait
        if(dataReceived) {
                transferAwait++;
        }
        // only if dataReceived is true and transferAwait is over 1000 allow
        // accessing to this condition
        if(dataReceived && (transferAwait > 1000)) {
                counter = 0;            // Reset variable
                transferAwait = 0;      // Reset variable
                dataReceived = false;   // Reset variable
                pc.println(gsmBuffer);  // Print buffer to pc
                for(int i = 0; i < 100; i++) {          // Reset buffer
                        gsmBuffer[i] = 0x00;
                }
        }
}
```

Once you have compiled and programmed your Arduino open serial terminal. Try sending AT+CGSN to gsm. If using arduino terminal select carriage return from the list. GSM should now answer with its IMEI number. Check for examples at NEOWAY M590 at command sheet. When this works play around with other command to get familiar with the module.

4.9.2 Send SMS

Lets open the datasheet and look up for reuired commands and configurations. For this example we need to edit previous code quite a lot. Let’s define that when we press o from keyboard system will turn on or off, when press c neoway will be configured and when s message is sent.

```
#define pc Serial
#define gsm Serial3
#define gsm_on 21
#define gsm_ring 22
char gsmBuffer[100];
char counter = 0;
bool dataReceived = false;
int transferAwait = 0;
int delayCounter = 0;
char receivedChar = ' ';                    // Memorize received character from PC
```

Setup stays same as in previous example.

```
void setup() {
                // put your setup code here, to run once:
                pc.begin(9600);
                gsm.begin(115200);
                pinMode(gsm_on, OUTPUT);
                pinMode(gsm_ring, OUTPUT);
                initializeGSM();
}
```

Loop will change quite a lot. When characters o, c or s is received from pc we check the received character with if condition. At first condition we check if receivedChar is equal to o. If this true then we toggle gsm modules power. Rest works in same manner.

After every loop receivedChar is cleared.

Function sendTestMessage determines the phone number to send the message and message to be sent. In this case message is hello.

```
void loop() {
  // put your main code here, to run repeatedly:
                pcToModule();
                if(receivedChar == 'o') gsmPowerToggle();
                if(receivedChar == 'c') configureDevice();
                if(receivedChar == 's') sendTestMessage("+33123456789", "Hello");
```

```
            receivedChar = ' ';
}
```

Function to toggle power. To toggle power gsm on pin has to pulled low for 300mS and then back high. When this function is called ”Toggling gsm power” is sent to terminal.

```
void gsmPowerToggle()
{
  pc.println("Toggling gsm power");
  digitalWrite(gsm_on, LOW);  // Stay
  delay(300); // Wait for 300mS
  digitalWrite(gsm_on, HIGH); // Power on system
}
```

This function clears buffer and toggles system power. When device is started this function tries to start the gsm module.

```
void initializeGSM()
{
  for(int i = 0; i < 100; i++) {
      gsmBuffer[i] = 0x00;
  }

  digitalWrite(gsm_on, LOW);  // Stay
  digitalWrite(gsm_ring, LOW);
  delay(300); // Wait for 300mS
  digitalWrite(gsm_on, HIGH); // Power on system
}
```

Here we configure the device to be able to send SMS messages.

AT+CMGF=1 sets gsm module to text format. 0X0D is hex for carriage return. Some of commands requires this as end character.

AT+CSCS sets gsm moduel to understand text as default alphabets.

```
void configureDevice() {
  pc.println("Configuring device");
  gsm.write("AT+CMGF=1");
  gsm.write(0x0D);
```

```
  delay(100);
  gsm.write("AT+CSCS=\"GSM\"");
  gsm.write(0x0D);
  delay(100);
}
```

When this is called message is sent to phone variables number and message is msg variable.

```
void sendTestMessage(String phone, String msg) {
  pc.println("Sending message");
  delay(200);
  gsm.print("AT+CMGS=\"" + phone + "\"");
  delay(500);
  gsm.write((char)0x0D); // Carriage return
  delay(500);
  gsm.print(msg);
  delay(500);
  gsm.write((char)0x1A); // Ctrl-z
}
```

This reads any incoming character from pc or gsm module and does as defined for them. If anything received from gsm module, this data is sent to PC. If data is coming from PC this incoming character is stored to receivedChar and processed in loop function.

```
void pcToModule()
{
  while(pc.available() > 0) {
    receivedChar = pc.read();
   // gsm.write(pc.read());
  }

  while(gsm.available() > 0) {
    transferAwait = 0;
    char inChar = ' ';
    inChar = gsm.read();
    gsmBuffer[counter++] = inChar;
    dataReceived = true;
  }
```

```
 if(dataReceived) {
   transferAwait++;
 }

 if(dataReceived && (transferAwait > 1000)) {
   counter = 0;
   transferAwait = 0;
   dataReceived = false;
   pc.println(gsmBuffer);
   for(int i = 0; i < 100; i++) {
     gsmBuffer[i] = 0x00;
   }
 }
}
```

Once you uploaded the code to arduino and checked that pins are correct. Open terminal and try configuring the device by sending ’c’ to arduino. If nothing is replied send ’o’ and see if anything is returned. If data is returned to any of the command process as follows.

First configure the device with ’c’ then send the message with ’s’. both of these should respond with ’OK’.

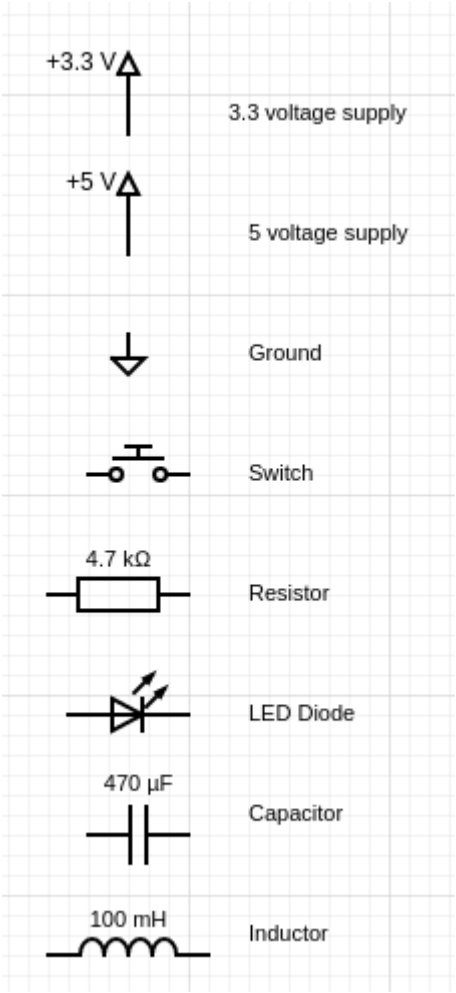
5 Circuit Design

5.1 Basics

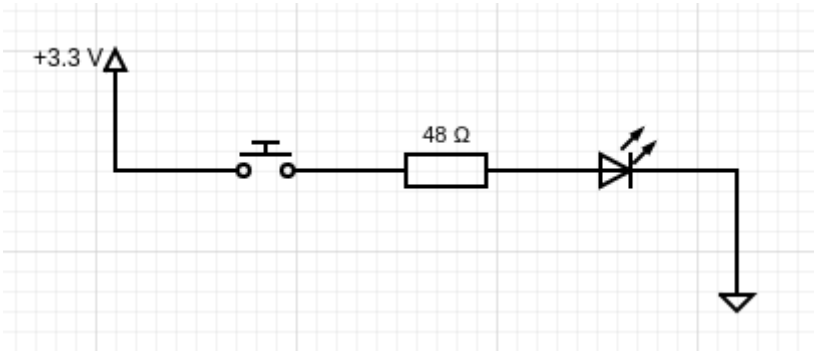
5.1.1 How to read schematics

To design circuits you need to be able to read schematics. Engineer makes a circuit design and then builds a circuit based on the design. Let’s say we are designing a switch that turns a led on when button is pressed. Schematic would look like this. To design a circuit you can either draw to a paper or use KiCad a free circuit design suite. For demonstration purposes I’m using circuit diagram. If starting I recommend drawing on paper.

You can see some basic symbols from symbol chart.



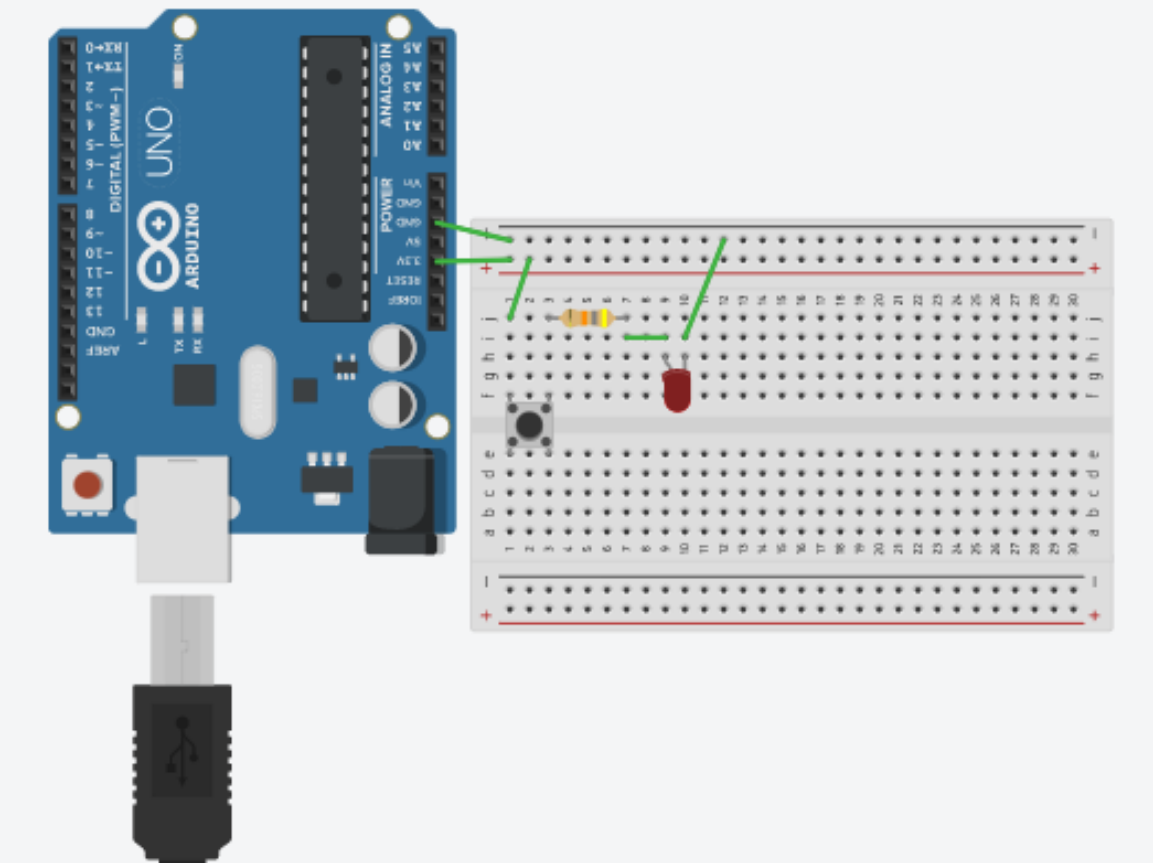
So to turn on LED we need a LED diode, a switch to turn it on and off. Resistor is usually needed to protect the diode. With resistor we can limit current flowing through LED diode. If too hig current flows through LED, it can brake. We need a 3.3V supply and a ground. The design would look something like this.



Black lines connecting each symbol represents wire. Resistor value is determined based on LED specifications. Lets assume LED Vf(forward voltage) is 2.1 and it’s max current is 30mA. I would drop the current a bit lets say 25mA to play it safe. Exceeding the max current can damage LED. Now we need to find out what is voltage over LED. Because LED will consume some of the voltage overal voltage over led is quite small. This can be calculated by Voltage in – Led forward foltage divided by current. U = I*R

→ R = U/I. For Led R = (3.3 – 2.1)/25mA = 48 Ohms. Resistor on front of LED is called LED current limiting resistor.

Now you have the circuit design finished so lets start making it on a test board. You can buy fairly affordable breadboards from Aliexpress. Search with term breadboard.



Your board should be now somewhat similiar, green lines represents wire. Arduino is used as power supply.

5.2 KiCad

KiCad is a free circuit design software for Windows and Ubuntu. Very usefull if you decide to create your own circuits.

5.2.1 Creating custom symbol for component

To create a symbol open from tools library editor. You can either create a new library or used existing. To use existing select a library where you want to add this component from File and select library. To create new one from File → save library as.

On left top you should see op-amp icon on right side of the book with magnificatio glass. Click that to create new component. On the right side you can see different tools to draw your symol. You can select pin atool underneath arrow tool. Once

finished saved your symbol. If you made a new one you need to add it tou your KiCad from preferences and component libraries.

5.3 Power Supply

When designing circuit on PCB it’s prefered to use Switching power supllies over linear regulators. Linear regulators tend to heat quite a lot if high currents are requires and effeciency is far from what switching supllies can provide.

Linear regulation working principle is quite well explained on Thomas Floyed book Electronics Devices Conventional current edition. Linear regulator has a control element, error detector and sample circuit. Reference voltage is driven to op-amps which determines what the output voltage should be. Op-amp controls an control element which is a transistor. If output voltage starts to decrease sample circuit provides a propotional voltage to op-amps inverted input. This difference between Vref and sample circuits voltage is then amplified to Control element. Control element will start increasing Vout until voltage to op-amps inverted input equals to Vref. Opposite action happens if output voltage tries to increase. Heatsink is usually attached to power transistor which is the control element. Transistor conducts most of the heat due most current flows through it.

Ref. 19

Advantages on linear regulator is it’s simplicity quite often just 3 pins are required. Vin, gnd and Vout. Because switching regulators works on high frequency linear regulators do not generate noise in same manner. Switching regulators output should always be filtered with capacitors and they need to be taken account when designing groun plate.

Switching regulators provide much more effeciency. In switching circuit the control element(transistor doesn’t) conduct constantly as in linear circuit. In linear circuit this constant current flow heats up the transistor heavily. Switching circuit turns on and off the control element in high frequency thus heat dissipation is lower.

Switching circuit charges and discharges the capacitor to generate wanted voltage. Charge and discharge period is determined by the duty cycle. On on-time charging and on off-time discharging. While Capacitor is used to store and maintain voltage level inductor is used to maintain current. When on on-time current flows to inductor and it stores it’s energy. When off-time hits inductor realeases it’s energy and rapidly loses it’s current to output. Ref. 19, 20.

5.4 Swithcing power supply: Buck converter simple

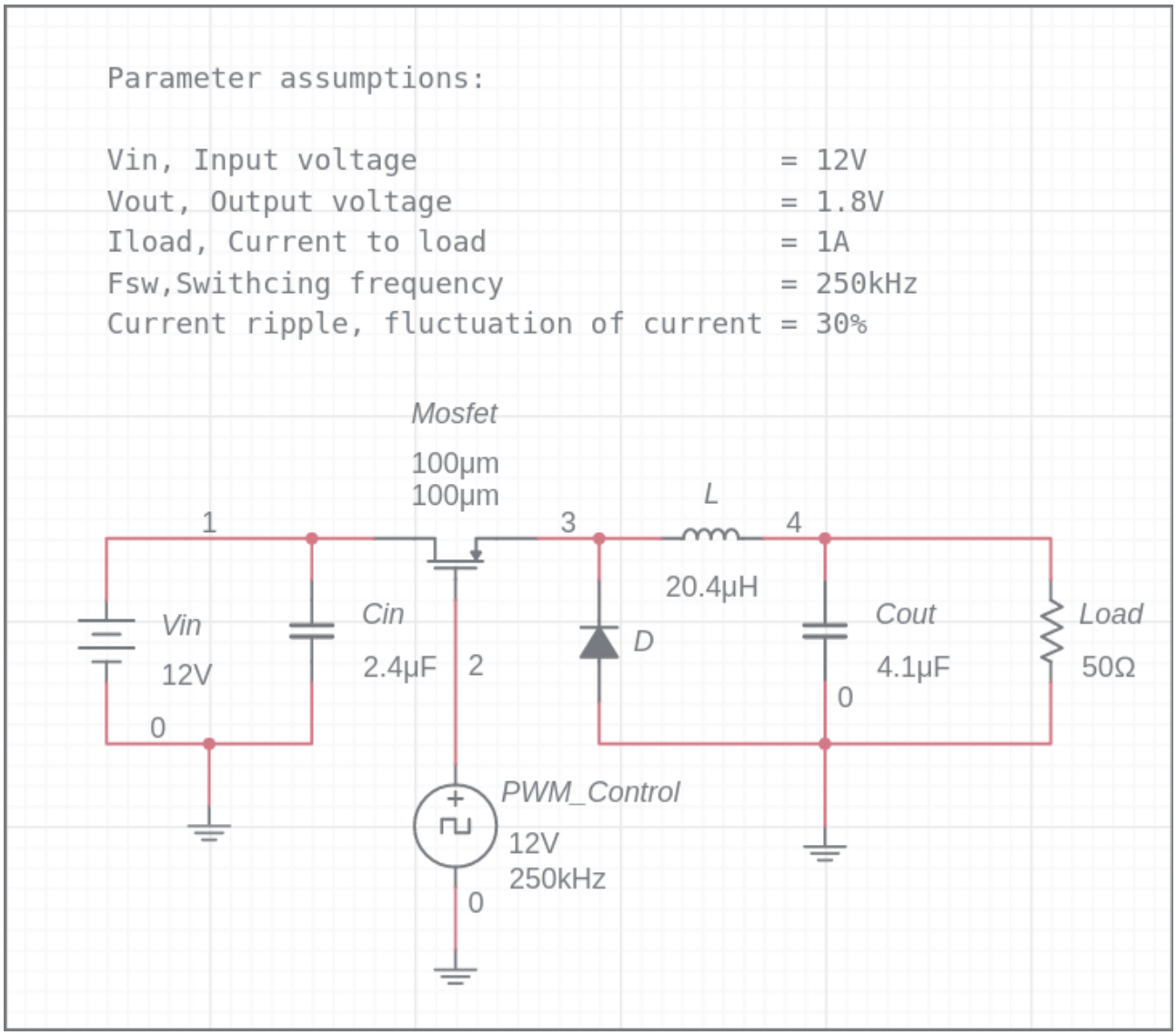


Illustration 10: Buck converter

Buck converter converts input voltage to lower voltage.

Determining inductance.

Inductor voltage: VL = L * di/dt.

L = V * (dt / di), V = (Vin – Vout).

L = (Vin – Vout) * ((D / Fsw) / Iripple), D = Vout / Vin.

!Because Load and resistance of buck converter is in series voltage division applies thus voltage over inductor is Vin – Vout.

D = 1.8V / 12V = 0.15

D / Fsw = 0.15 / 250 * 10^3 = 6 * 10^-7 = 0.0000006

Iripple = 30% * Iload = 0.3 * 1A = 0.3A

L = (12V – 1.8V) * ((6*10^-7)/ 0.3A)

L = 10.2V * 2*10^-6 = 10.2V * 0.000002

L = 20.4uH → 0.0000204H

Power dissipation = (Iripple)^2 * ESR, ESR resistance of inductor, see datasheet.

Lets assume our inductors ESR os 0.02ohms.

P = (Iload)^2 * ESR

P = 1^2 * 0.02

P = 20mW = 0.02W

Inductor inductance is 20.4uH and Power dissipation 20mW.

Wats comes in use later on determining effeciency of buck converter.

Determining output capacitor Cout

ΔV= ΔI * (ESR + ΔT / C)

Define allowed ripple voltage : 50mV

ΔI = Iripple = 30% * Iload= 0.3A

Check ESR from selected capacitors datasheet. Let's assume it's 0.02ohms.

ΔT = D/Fsw = (Vout / Vin) / Fsw

ΔT = (1.8V/12V) / 250kHz

ΔT= 600nS = 0.6uS

ΔV = ΔI * (ESR + ΔT / C) → C = (ΔI * ΔT) / (ΔV - (ΔI * ESR)), ΔV = Ripple voltage = 50mV

C = Cout = (0.3A * 0.6uS) / (0.05 -(0.3A * 0.02))

C = (1.8 * 10^-7) / (0.044)

C = 4.1uF. This is minimium value for output capacitor.

Power dissipation = (Iripple)^2 * ESR, current flowing through capacitor.

Power dissipation = 0.3^2 * 0.02 = 1.8mW

Determining Input capacitor Cin

Estimate input ripple current. Worst case ripple current occurs when duty cycle is 50% and the worst case ripple current on the input is about half of load.

!Often input Iripple is higher than output Iripple thus Iripple for input → Iripple = Iload/2.

Iripple = 1A/2 = 0.5A

Define accapteable input ripple voltage. Let's assume 200mV.

Select capacitor acceptable ESR value. We assume it to be 0.15 ohms.

C = Cin = ΔT / ((Vripple / Iripple) – ESR). ΔT = D/Fsw, D = Vout/Vin

ΔT = (Vout / Vin) / 250*10^3

ΔT = 0.6u → 0.6 * 10^-6

C = 0.6u / ((0.2/ 0.5) – 0.15)

C = 2.4uF

Power dissipation = (Iripple)^2 *ESR

Power dissipation = (1/2)*0.15

Power dissipation = 0.075W → 75mW

Determining diode D

Estimate diode current Id,

Id = (1 – D) * Iload, D = Vout / Vin

Id = (1 – 0.15) * 1A

Id = 0.85A

Max diode reverse voltage is Vin thus 12V

Power dissipation = Vf * Id, Vf can be found from datasheet of schottky diode. Vf = Forward voltage drop. Check from scacle which Vf is closest to Iload.

Lets assume Iload is 0.3V@1A

Vf = 0.3*0.85

Vf = 255mW

Determining Mosfet.

Vin = 12V, Iload = 1 A, Vout = 1.8V, Vin = 12V, Fsw = 250kHz.

Trise and Tfall check from mosfet Datasheet but lets assume acceptable is 55nS for both

Trise = 55nS

Tfall = 55nS

Select Mosfet with low RDS. Low RDS minimizes generated temperature.

Ensure VGSmax is higher than Vin to endure voltage peaks.

RDS → Lets assume to be 0.02ohm

Coss → can be found from datasheet. In this case lets assume 500pF.

Pconduction = (Id)^2 * RDS * D

D = Vout / Vin

D = 1.8 / 12 = 0.15

Id = 1, equal to peak current current through mosfet

Pconduction = (1)^2 * 0.02 * 0.15 = 3mW

Pswitching = (V * Id / 2) * (Ton * Trise) * Fsw + (Coss * V^2 * Fsw)

V = voltage over mosfet = Vin - Vout

Pswitching = (10.2V * 1/2) * 110ns * 250kHz + (500pf*10.2^2*250kHz)

Pswitching = 5.1 * 110ns * 250kHz + 127.5

Pswitching = 140mw

Caculating effeciency

Input capacitor loss:	75mW
Output capacitor loss:	1.8mW
Inductor loss:	20mW
Diode loss:	255mW
Mosfet:	140mW
Total losses	492mW

Total output power → P=U*I → P=Vout * Iload = 1.8W

Efficiency = 1.8W/(1.8W + 492mW) = 0.79 %.

This became fairly poor effeciency. Effeciency aim should be over 90%.

ref. 21

5.5 Swithcing power supply

In this example I'm using ACT4060A. Due to it's cheap from aliexpress and quite powerfull. It cand drop voltage from 4.5 – 24V, can output 2A and has adjustable output. I'm going to show here one 3.3V whit KiCad. If you open datasheet and page 8, on figure 4 there's quite good example.

You might need to create a new symbol and footprint for this. This process is shown in KiCad chapter.

Ref. https://batteryuniversity.com/learn/article/charging_the_lead_acid_battery

Ref. https://haynes.com/en-us/tips-tutorials/what-lead-acid-battery-your-car

Ref. https://www.batterystuff.com/kb/articles/battery-articles/battery-basics.html

5.6 1-Cell chargher design for Li-Ion / Li-Polymer Charge

Charge management controllers are premaid IC chips and fairly cheap. Without proper battery management lithium batteries can be very dangerous. If heavily over charging a 1-Cell lithium battery will very much burst into fire. Ready made chips provide

excellent charge management. Very highly recommended to use well known manufacturers chips.

I’m using here MCP73833 from Microchips. Chip is very affordable and can be soldered to PCB by hand. Most charge management chips are in TTSOP or DFN package thus quite hard to solder and hard to find physical faults. MSOP-10 package is still manageable with hands and easy to solder to ensure best possible functionality. There is also MCP73831/2 chip with SOT-23-5 package wchis is very easy to solder. Also a bit simpler than MCP73833.

Check first that this design is suitable for your battery type and for your hardware design from datasheet. I will use KiCad to make this design and implement it to my project.

Start by checking the typical application from the datasheet

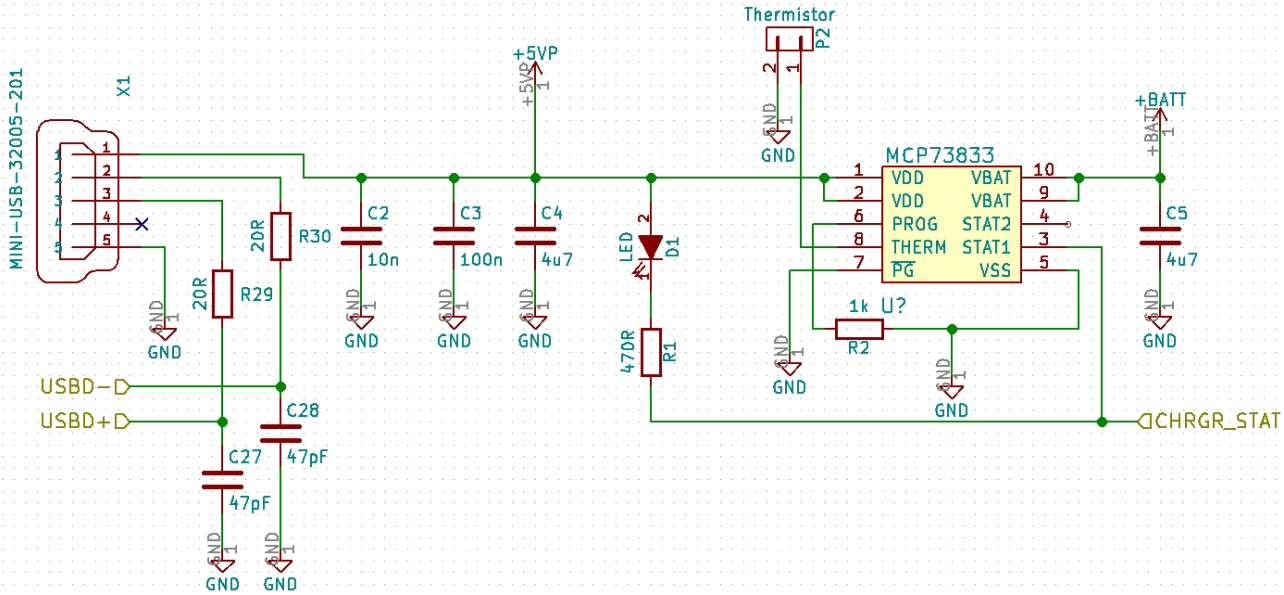


Illustration 11: MCP73834

X1 represents Micro USB port. Pin 1 is 5V line and pin 5 ground. Pin 2 and 3 here goes straigt to microcontrollers programming port but that is irrelevant to this example.

C2 and C3 are quite small capacitors they are chosen because smaller capacitors are faster to discharge thus reacting faster to voltage change. C4 is main input capacitor and meant to balance heavier fluctuation at the line. Having multiple capacitors in parralel will also decrease their ESR value. ESR is the resistance of capacitance and this can cause power losses.

Ref. 22

C5 provides compensation when there is no battery load and improves output voltage stability. Program pin is pulled to ground throught 1k resistor. 1K resistor tells the unit

to set charging current to 1Amp. Equation for this is Ireg = 1000V/Rprog. → 1A = 1000V/1000Ohms.

When chip starts charging the battery STAT1 pin is pulled down and will behave as a ground. This will turn on the LED D1. CHRGR_STAT here is connected straight to MCU to know when the unit is being charged. In case you have a program that show battery level to user, charge voltage will give false information to user. When unit starts charging the battery voltage will be risen depending on the resistance of battery to provide the needed charghing current. Once charge is completed STAT1 will go high and Led dims.

PG pin in this case is TE pin. More info why from datasheet. TE pin pulled low chips internal timer is activated. This timer will shutdown the chip after certain time. More details at datasheet.

Therm pin is connected to 2-pin JST connector. From the connection a thermistor is pulled on the battery to provide battery temperature monitoring. If battery voltage rises charging will stop. If battery starts malfunctioning it will usually start heating and expanding.

VBAT pin is pulled to plus of the lithium battery and lithium batterys negative pin to ground.

On illustration 10 P1 is connected to battery. +BATT source is connected to the VBAT of charging circuit. Fuse is quite badly located straight front of battery. It would be fine if it would be only source to provide voltage to the system but this is not the case. More of this later. Voltage divider provides acceptable voltage to microcontrollers analog in pin.

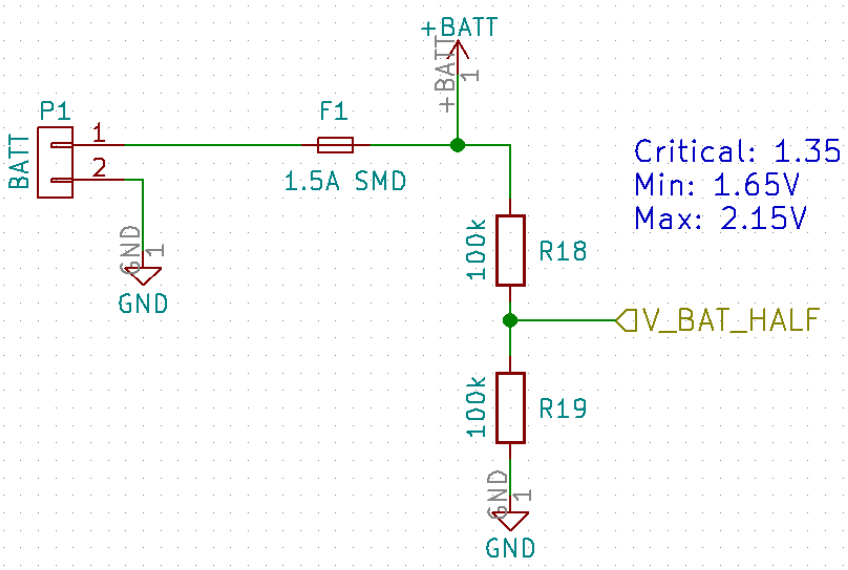


Illustration 12: Battery voltage level reading

The circuit on Illustration 10 is to seperate battery from load while charging. I can be wrong but personnaly I want to seperate the battery to ensure the charger has absolut control and correct value of battery voltage level.

If load is active this can lower the voltage level of battery from what it actually is because high current need. This would lead to false information to chargher and charger would always try to charge the lithium battery to 4.2 volts. This could lead over charghing the circuit and dropping the life span of battery or in worst case depending on the condition of battery lead to battery realeasing fumes and bursting to fire. I can be wrong here but prefer to take safest road.

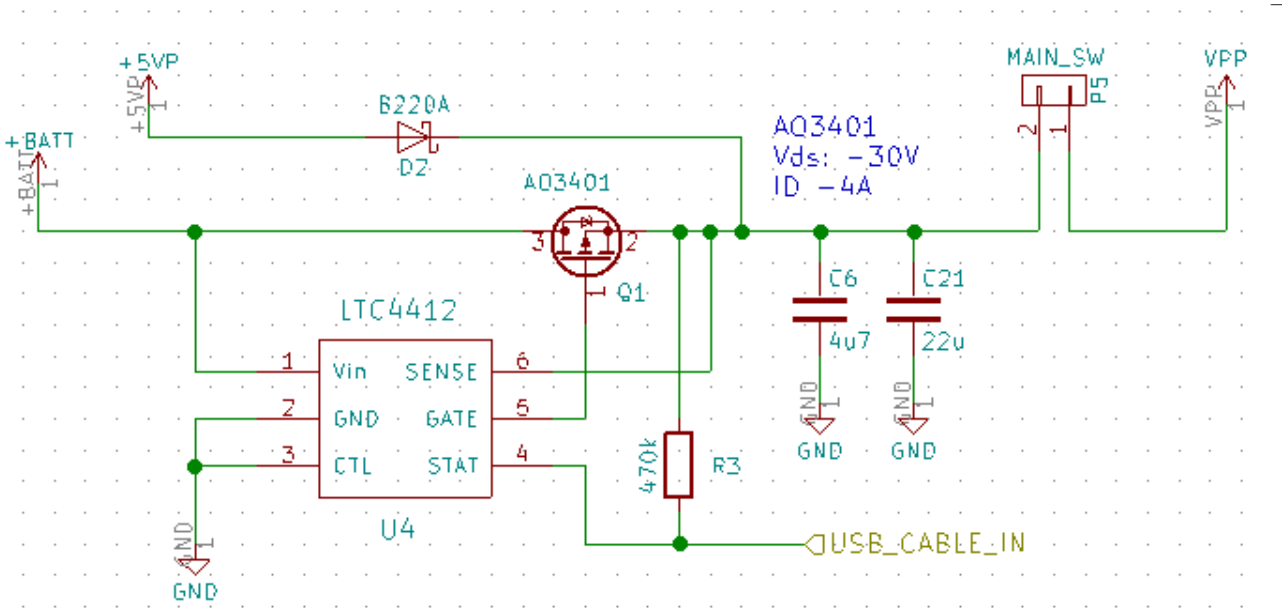


Illustration 13: LTC4412

When USB is plugged in to the USB port +5VP will go high and LTC4412 will detect this. This will lead for closing the mosfet AO3401 thus seperating battery from load. Load will still steady voltage from the USB. Current provided by USB will be divided by charging and running the circuit. The MAIN_SW is a main power switch to disconnet or connect load with the power supply. Main switch is set after the power supply to allow charging without holding system on.

LTC4412 compares Vin and Sense pin. When SENSE pin voltage is higher than VIN chip will pull p-channel mosfet Q1 high thus disconnecting battery.If voltage in SENSE is lower than Vin – 20mV battery will be connected to load. Voltage rises to 4.3V when USB is plugged at SENSE pin and whne battery is conducting voltage will be around 3.3 to 4.2. STAT pin will tell microcontroller if USB is plugged or not.

6 Circuit Protection

When building circuits it's always good to have protection systems monitoring your circuit. Here we concentrate on battery oriented systems. This protection unit has been built to protect the customer/user and “load” itself. Here we go through Over-voltage, over-current/short-circuit and over-discharge.

6.1 Over-voltage protection

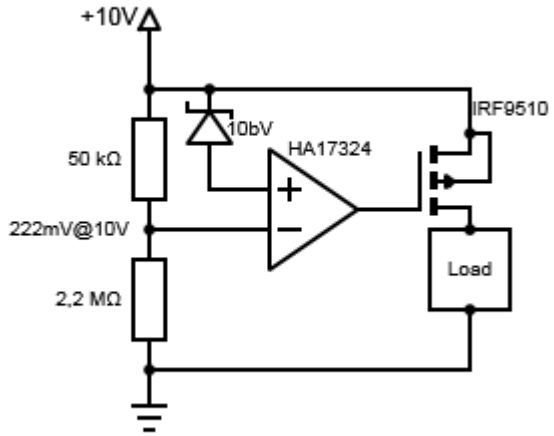


Illustration 14: Over-voltage protection

At the illustration (5) we can see one method to protect your circuit from over-voltage by simple op-amp and mosfet circuit. Circuit brakes loads positive side(+10V) off when voltage goes higher than 10 volts. The comparator operational ampilfier circuit hold mosfets gate voltage low. Comparator operation amplifier compares voltages of opamp + and opamp -. When opamp + goes higher than the opamp – voltage, opamps output will turn high. The zener diode starts conducting voltage gradually when positive side reaches 10V. If postive side goes higher than 10V, voltage through zener diode increases and exceeds reference voltage(~222mV), thus turning opamps output high.

When opamps output turns high mosfet will stop conducting, thus cutting loads positive supply. The pchannel mosfet IRF9510 needs gate to source voltage difference of ~-10 to fully conduct. So if this mosfets source voltage equals to gate voltage, mosfet wont conduct.

6.2 Over-current protection

With over-current protection we protect the circuit from shorts or “decaying components”. Method used in this tutorial analyses voltage across resistor with operational amplifier. We use shant resistor at positive side of load to generate small voltage drop on load.

Lets go through basics.

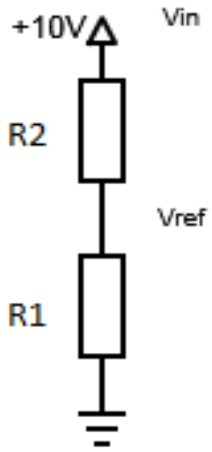


Illustration 15: Voltage divider

Vin	10
R2(kOhms)	2200

R1(kOhms)	Vref	I(mA)
2200	5	2,27
2000	4,76	2,38
1800	4,5	2,5
1600	4,21	2,63
1400	3,89	2,78
1200	3,53	2,94
1000	3,13	3,13
800	2,67	3,33
600	2,14	3,57
400	1,54	3,85
200	0,83	4,17

Illustration 16: Current_detector

In basic voltage divider the Vref depends on ratio of R2 and R1. Lets say R2 is 2,2 mega Ohms and R1 is varying. Vref voltage across R1 is calculated with **Vin(+10V)*(R1/(R1+R2)**).As you can see from illustration (7), lower resistance means lower voltage and lower resistance higher current. So basically the higher the current flowing through R1 and R2, lower the vref will be across R1.

By using operation amplifier comparator circuit we can detect “loads” current through voltage change.

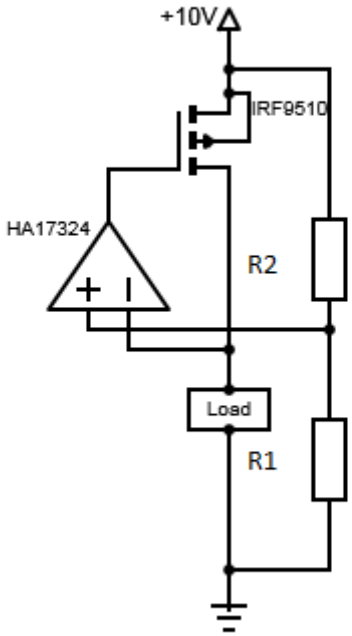


Illustration 17: Current detector

In illustration (8) we have current detector, this circuit will cut-off current flow to load if current flow increases too high. With R2 and R1 we can determine cut-off current. When voltage across R1 is lower than voltage across load, the operational amplifiers output stays low. If current flow through load increases, voltage across load starts decreasing. When voltage across load drops below voltage across R1, operational amplifiers output turns high and p-channel mosfet stops conducting.

6.3 Over-discharge protection

When dealing with batteries to ensure effecient use and avoid the battery itself damaging, discharge protection is mandatory. For example lithium batteries needs to operate inside certain voltage range. These lithium batteries builds from lithium cells and each cells recommended operational voltage is around 3.3V to 4.2V. If we had two cell lithium battery, our batterys operational voltage would be 2 * 3.3 – 2 * 4.2 → 6.6V to 8.4V. With illustration (9) circuit we can limit its current flow to 6.6V. At this point we dont have to care of max voltage 8.4V because this is usually limited with battery charger.

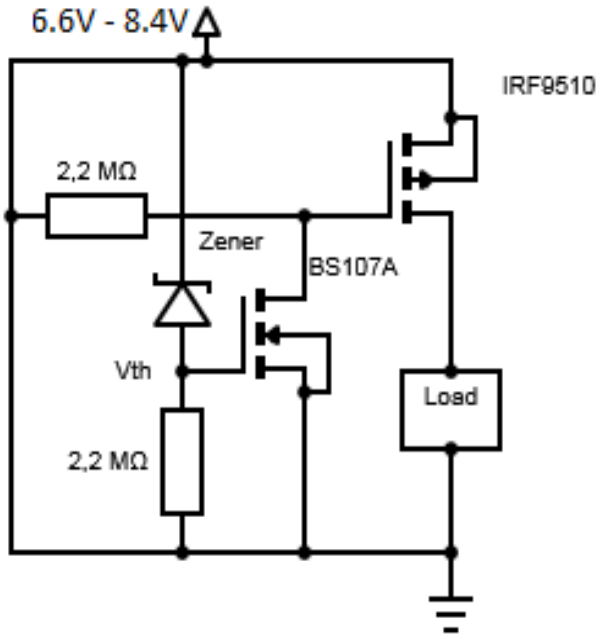


Illustration 18: Discharge_Protection

This circuit will only conduct to load, when battery voltage is over 6.6V. When voltage is over 6.6V BS107A mosfet pulls IRF9510 mosfet to gnd. While IRF9510 gate is pulled to gnd, load will be operational. When voltage drops under ~6.6V BS107A will stop pulling IRF9510 to gnd and current flow to load stops. This is not an accurate 6.6V system because of BS107A characteristic. BS107A requires ~1.3V to 2V to start conducting. The threshold voltage needed for BS107A depends on current of drain to source(Ids) . Higher the current needed to pulled through, the higher Vth is needed.

Because of mosfet characteristic, it will take some time for IRF9510 to be totally open. Mosfets resistance depends on difference source and gate voltage. In case of IRF9510 gate should be -10V from source voltage to be totally open. If mosfet is not totally open(at saturated state) it will still have higher resistance than in saturated state thus limiting current flow through the mosfet.

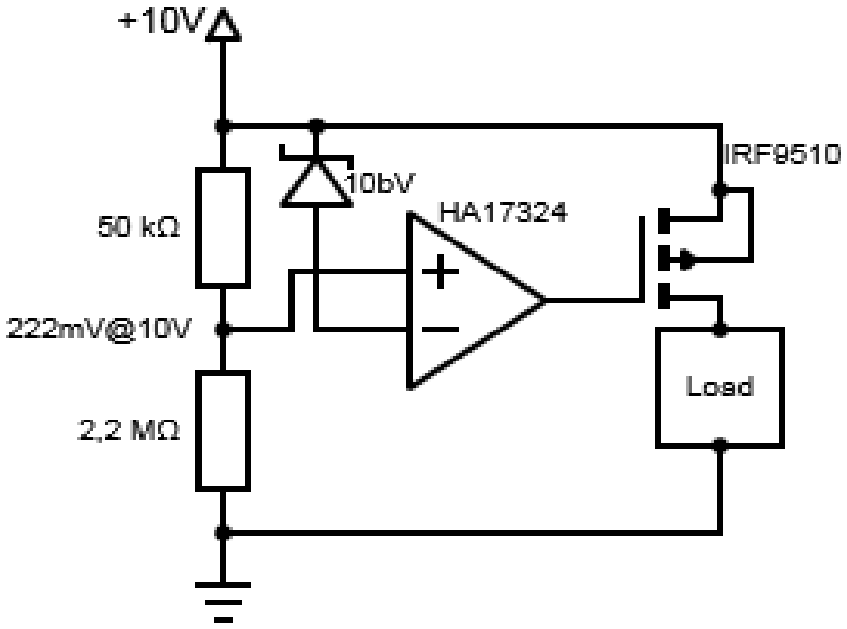
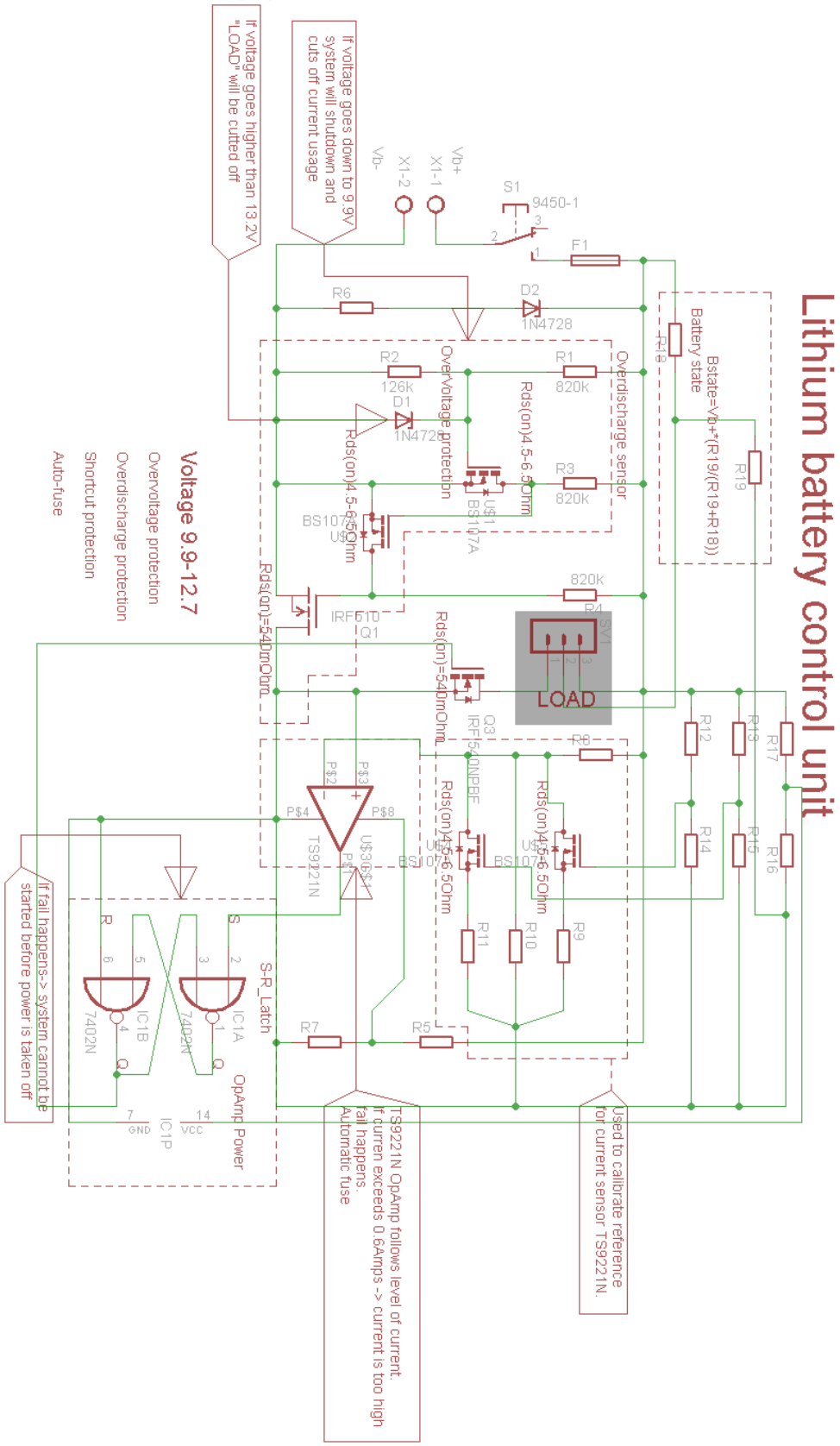


Illustration 19: Over_Discharge_Opamp

Another method would be using operational amplifier to detect threshold voltage. This circuits proofs to be more accurate. We can determine the threshold voltage with zener diode(opamp-) and voltage divider (opamp+). Illustration (10) circuit will start conducting to load when voltage passing through zener reaches to 222mV.

6.4 Lithium battery protection circuit



Components:

S1	:	9450-1
F1	:	FUSE
R6	:	2.2M
D2	:	1N4728
R1	:	820K
R2	:	126K
D1	:	1N4728
R3	:	820K
U$1	:	BS107A
U$2	:	BS107A
Q1	:	IRF510
Q3	:	IRF540
SV1	:	2.54mm HEADER
R17	:	557R
R13	:	820K
R12	:	820K
R16	:	391R
R15	:	112K
R14	:	103K
R8	:	3235R
Q5	:	BS107A
Q6	:	BS107A
U$3	:	TS9221N
R9	:	1105R
R10	:	108R
R11	:	1054R
IC1A	:	7402N
IC1B	:	7402N
R5	:	328R
R7	:	329.1R

7 Jetson

7.1 Flash

Download Jetpack 3.3 or newer from NVIDIA
https://developer.nvidia.com/embedded/dlc/jetpack-l4t-3_3

open terminal in same folder as jetpack

Type to terminal chmod +x jetpack-l4t-xx

install xterm → sudo apt-get install xterm

run jetpack ./jetpack-l4t-xx

If warned of OS → press continue

follow instruction, instruction are quite self explaining.

7.2 Set SPI

Reference https://elinux.org/Jetson/TX2_SPI

- git clone http://github.com/jetsonhacks/buildJetsonTX2Kernel.git
- cd buildJetsonTX2Kernel
- ./getKernelSources.sh
- cd *usr*/src/kernel/kernel-4.4/arch/arm64/configs/
- sudo gedit tegra18_defconfig
- add CONFIG_SPI_SPIDEV=m as follows
  - CONFIG_SPI=y
  - CONFIG_SPI_TEGRA114=y
  - CONFIG_SPI_SPIDEV=m
  - CONFIG_QSPI_TEGRA186=y

Build kernel

- cd *usr*/src/kernel/kernel-4.4
- sudo make tegra18_defconfig
- `cd ~/buildJetsonTX2Kernel`
- `sudo ./makeKernel.sh`
- Check from */usr/src/kernel/kernel-4.4/drivers/spi/ that you have file spidev.ko if this doesn't exist re-make kernel without sudo*
- *sudo cp /usr/src/kernel/kernel-4.4/drivers/spi/spidev.ko /lib/modules/$(uname -r)/kernel/drivers/*
- *sudo depmod*

- *sudo ./copyImage.sh*
- *sudo reboot*
- *Verify from lib/modules/$(uname -r) modules.dep file that you can see kernel/drivers/spi/spidev.ko: kernel/drivers/spi/spidev.ko*
- *If you can only see kernel/drivers/spi/spidev.ko: modify this to look as on top.*
- *Also check from lib/modules/$(uname -r)/kernel/drivers/spi/ spidev.ko file exist*
- *Install DTC tool*
- *sudo apt-get update && sudo apt-get install device-tree-compiler*
- *cd boot/dtb/*
- *sudo dtc -I fs -O dts -o extracted_proc.dts /proc/device-tree*
- *sudo gedit extracted_proc.dts*
- *add following as follows:*
-

spi@3240000{

```
    compatible = "nvidia,tegra186-spi";
    reg = <0x0 0x3240000 0x0 0x10000>;
    ....
    ....
    ....
    linux,phandle = <0x80>;
    spi@0 {
      compatible = "spidev";
      reg = <0x0>;
      spi-max-frequency = <0x1312D00>;
      nvidia,enable-hw-based-cs;
      nvidia,cs-setup-clk-count = <0x1e>;
      nvidia,cs-hold-clk-count = <0x1e>;
      nvidia,rx-clk-tap-delay = <0x1f>;
      nvidia,tx-clk-tap-delau = <0x0>;
    };
 };
```

Recompile the device tree

- *cd boot/dtb/*
- *sudo dtc -I dts -O dtb -o tegra186-quill-p3310-1000-c03-00-base.dtb extracted_proc.dts*

Enable the new dtb

- *cd boot/extlinux/*
- *sudo gedit extlinux.conf*
- *make sure it is as follows*

```
 TIMEOUT 30
 DEFAULT PRIMARY
 MENU TITLE p2771-0000 eMMC boot options
 LABEL primary
     MENU LABEL primary kernel
     LINUX /boot/Image
     FDT /boot/dtb/tegra186-quill-p3310-1000-c03-00-base.dtb
     APPEND ${cbootargs} root=/dev/mmcblk0p1 rw rootwait rootfstype=ext4
```

- *sudo reboot*
  Test that all works
- *ls /dev/spi* you should see spidev.3.0*

7.3 UART communication HM-10 BLE

Connect HM-10 BLE to raspberrry. In case of HM-10 with brakeout board connect HM-10 VCC to 5V otherwise to 3.3V. TX and RX to UART1 TX and UART01 TX. HM-10 commands and datasheet http://www.martyncurrey.com/hm-10-bluetooth-4ble-modules/

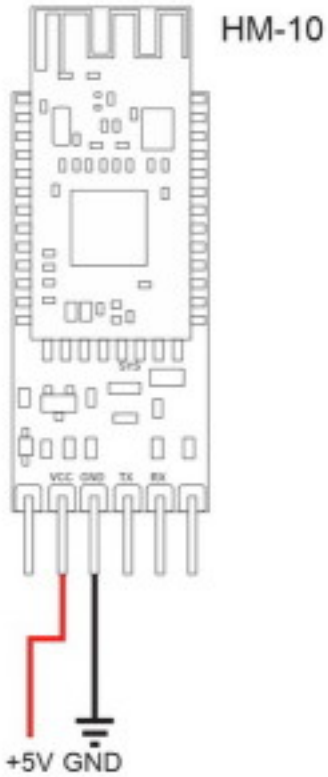


Illustration 20: http://www.martyncurrey.com/hm-10-bluetooth-4ble-modules/

Datasheet: http://fab.cba.mit.edu/classes/863.15/doc/tutorials/programming/bluetooth/bluetooth40_en.pdf

Turn on Jetson. Open minicom settings → minicom -s. Ensure baudrate is 9600 and port is ttyTHS2(UART1).

To test connection enable local echo by pressing cntrl + A and then e. Type AT and press enter. HM-10 should now reply with OK message. If no OK is replied check your wiring.

Make a file uarthHM10.py(python required) and copy following code.

Device address ttyTHSx can be different, than in example.

```
Import serial


with serial.Serial('/dev/ttyTHS2', 9600, timeout = 1) as ser:
        print(ser.name)
        ser.write("AT")
        line = ser.readline()
        print(line)
```

Exit editor and run the file "python uartHM10.py" If all goes correctly HM10 should reply you "OK". Let's change name of the unit to Jetson_ble. Helps finding device.

Add to previous code following (indentation must follow previous code. This code continous with same indentation as previous code print(line) and rest. Python is very picky on this)→

```
        ser.write("AT+NAME=raspberrry_BLE")
        line = ser.readline().
        print(line) # This should print "OK"
        ser.write("AT+NAME?") # Query to get ble name
        line = ser.readLine()
        print(line) # This should now print raspberrry_BLE.
```

Lets set HM-10 to discovery mode:

```
        ser.write("AT+IMME1")
        ser.readline()          # For some reason this is required
        ser.write("AT+ROLE1")
        ser.readline()
        ser.write("AT+DISCS")  # Start discovery
        ser.readline()
        ser.write("AT+DISC?")
        while True:
```

```
                line = ser .readline()
                print(line)
                if ”OK+DISCE” in line:
                        print(”Re-discover”)
                        ser.write(”AT+DISCS”)
                        ser.readline()
                        ser.write(”AT+DISC?”)
```

8 Software programming

8.1 Django rest framework on EC2 instance

Django REST framework is python based REST framework. It’s quite useful for managing databases because it’s allowing usage of python. Django allows you to manage your database trough admin view.

8.1.1 Django basics

8.1.1.1 Importing modules with same name

If having multiple modules with same name and you need to import them on same module/file, you can define different names.

For example importing two modules with name views

```
from project.app import views
from rest_framework.authtoken import views as authviews
```

You can now use views from your app as views.module and rest_framework views as authviews.module.

8.1.1.2 Script to run and combile your Django

If you want to make it easier to start server you can build a scrips to help here.

On the same folder as you have your manage.py make a script.

```
$vim runServer.sh
```

Add following

```
python manage.py makemigrations
python manage.py migrate
python manage.py runserver private_ip:port
```

Give the rights to run for your script by

```
$chmod +x runServer.sh
```

Now you can simply start your server by the script and it will handle all necessary for you.

```
$./runServer
```

8.1.2 Starting and accessing EC2 instance

This instructs how to start a server compatible with Django framework.

If you don’t have a user create one at https://aws.amazon.com/ otherwise just login.

1. Select services from top left and select EC2
2. Select running instances and launch instance.
3. Select Ubuntu Server 18.04 and T2.micro if you are elgible for free tier. Otherwise recommend picking T2.nano instance. Click select and launch.
4. Review your selection and click launch. Create and download key pair. This is used to login to your instance. Launce instance after downloading key pair.
5. Make following script to same folder as you downloaded the key pair.

   ssh -i “your_file.pem" ubuntu@your_public_instance. You can copy the public instance url from AWS instance management. From management select your instance and copy the public dns address. Your script should look something like this ssh -i “security.pem" ubuntu@ec2-xx-xxx-xxx-xxx.eu-west-3.compute.amazonaws.com
6. Open terminal in your .pem and script folder and give correct right for your script chmod +x <file_name>. For .pem file chmod 400.
7. Run the script and you should have entered your instance.

Django REST framework is python based REST framework. It’s quite useful for managing databases because it’s allowing usage of python. Django allows you to manage your database trough admin view.

This instructs how to start a server compatible with Django framework.

If you don’t have a user create one at https://aws.amazon.com/ otherwise just login.

8. Select services from top left and select EC2
9. Select running instances and launch instance.
10. Select Ubuntu Server 18.04 and T2.micro if you are elgible for free tier. Otherwise recommend picking T2.nano instance. Click select and launch.
11. Review your selection and click launch. Create and download key pair. This is used to login to your instance. Launce instance after downloading key pair.
12. Make following script to same folder as you downloaded the key pair.

    ssh -i “your_file.pem" ubuntu@your_public_instance. You can copy the public instance url from AWS instance management. From management select your instance and copy the public dns address. Your script should look something like

this ssh -i “security.pem" ubuntu@ec2-xx-xxx-xxx-xxx.eu-west-3.compute.amazonaws.com

13. Open terminal in your .pem and script folder and give correct right for your script chmod +x <file_name>. For .pem file chmod 400.
14. Run the script and you should have entered your instance.

8.1.3 Installing Django REST framework

Start by installing virtual env and pip.

Update your ubuntu repository	$sudo apt update
Install python3 pip	$sudo apt install python3-pip
Install virtualenv	$sudo apt install virtualenv
Create virtual environment folder	$virtualenv -p /usr/bin/python3 venv
Activate virtual environment	$source venv/bin/activate
Install Django	$pip install djangorestframework
	$pip install markdown
	$pip install django-filter

8.1.4 Create a project

Create a project	$django-admin starproject <project_name> .
	$cd <project_name>
	$django-admin startapp rest
	$cd ..
Migrate current database	$./manage.py makemigrations
	$./manage.py migrate
Create superuser	$./manage.py createsuperuser

8.1.5 Creating first application

Serializers allow data such as querysets and model instances to be converted to native Python datatypes that can be then easily rendered into JSON, xml or other content types. Serializers also provide deserialization, allowing data to be converted back into complex types, after first validating the incoming data.

Ref. https://www.django-rest-framework.org/api-guide/serializers/

Create a serializer.py $vim project/app/serializers.py

Edit serializer as follows

```
from django.contrib.auth.models import User, Group
from rest_framework import serializers

class UserSerializer(serializers.HyperlinkedModelSerializer):

    class Meta:
        model = User
        fields = ('url', 'username', 'email', 'groups')

    class GroupSerializer(serializers.HyperlinkedModelSerializer):
        class Meta:
            model = Group
            fields = ('url', 'name')
```

Edit views $vim project/app/views.py

Add following

```
from django.shortcuts import render

# Create your views here.

from django.contrib.auth.models import User, Group
from rest_framework import viewsets
from tutorial.quickstart.serializers import UserSerializer,
GroupSerializer

class UserViewSet(viewsets.ModelViewSet):
    """
    API endpoint that allows users to be viewed or edited.
    """
    queryset = User.objects.all().order_by('-date_joined')
    serializer_class = UserSerializer

class GroupViewSet(viewsets.ModelViewSet):
```

```
    """
    API endpoint that allows groups to be viewed or edited.
    """
    queryset = Group.objects.all()
    serializer_class = GroupSerializer
```

Next we edit the main URL parser. This file directs url request to correct path. For example your app and admin view has url module. To access these you need to define addresses to the URL module.

Edit main url module $vim project/urls.py

```
from django.contrib import admin
from django.urls import path

from django.urls import include, path
from rest_framework import routers
from messii.rest import views

"""
REST framework adds support for automatic URL routing to Django, and
provides
you with asimple, quick and consistent way of wiring your view logic
to a set of URLs.
"""
router = routers.DefaultRouter()
router.register(r'users', views.UserViewSet)
router.register(r'groups', views.GroupViewSet)

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api-auth/', include('rest_framework.urls',
namespace='rest_framework'))
]
```

ref. 16

Before runing include rest_framework modul to Djangos installed apps.

Open settings.py with vim $vim project/settings.py

Add following to Installed_aps

```
INSTALLED_APPS = {
        ...
        ‘rest_framework’,
}
```

Under installed apps add paginations. They allow yo to control how many objects per page are returned. Following allows only 10 object per page returned.

```
REST_FRAMEWORK = {
  'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',
  'PAGE_SIZE': 10
}
```

Now lets do quick test for the API

Run the server $python manage.py runserver <EC2 instance pirvate IP:port>

Example $python manage.py runserver <172.31.34.24:8000>

You can see your private ip address from EC2 dashboard by selecting your instance. Private IP should be shown at the bottom view. Port is up to you to decide.

You need to run this in private address otherwise your public ip won’t direct you to Django server. Not sure why.

Open postman https://www.getpostman.com/ and send test request.

To send test request configure the request as follows.

Set postman request to GET.

To url field add ec2_instance_public_ip:port/users/

Open Header tap and set to key Content-Type and to value field application/json.

Send your request and Django should block this. You should get to your server console(terminal)

```
Invalid HTTP_HOST header: 'ec2-35-180-242-178.eu-west-3.compute.amazonaws.com:8000'. You may need to add 'ec2-35-180-242-178.eu-west-3.compute.amazonaws.com' to ALLOWED_HOSTS.

Bad Request: /users/

[09/May/2019 05:55:32] "GET /users/ HTTP/1.1" 400 61864
```

Add the indicated address to you project/settings.py file. In my case address to add is 'ec2-35-180-242-178.eu-west-3.compute.amazonaws.com'. Your allowed hosts should now look something like this

```
ALLOWED_HOSTS = [
```

```
    'ec2-35-180-242-178.eu-west-3.compute.amazonaws.com'
]
```

Save changes and start your server. This time you should see on the terminal following

```
Not Found: /users/
[09/May/2019 06:03:49] "GET /users/ HTTP/1.1" 404 2107
```

This means your server is now online and works. You can also connect trough browser, this wil lshow your main url file paths. As said on debug message users path is not found we need to add it to our main URL file.

Open your main url file with txt editor $vim project/urls.py

Add following under urlpatterns

```
 path('', include(router.urls))
```

Start server and send the request. You should now receive json data containing url, username, email and created groups. You can also connect trough browser and have graphical view.

You can either use POST to create new user or GET to retrieve all users created.

To Post change GET to POST and add to body the user data you want to create.

Also set body data to raw and ensure JSON(application/json) is set. Add following and send it.

```
{
    "username": "test",
    "email": "test@test.fyi",
    "groups": []
}
```

Confirm that this has indeed stored to server by changing back to GET request and send it. You should now see your user at results returned to GET request

8.1.6 Adding token authentication

This gives each user you create a token authentication. Each time user uses any of REST endpoint except creating user authentication is required. This authentication token is added to your request header and the token key is generated by Django.

Start by adding authtoken to your installed apps.

$vim project/settings.py

Add following under installed apps.

```
INSTALLED_APPS = (
                …
                ‘rest_framework.authtoken’,
                ‘project.app’,
)
```

Under rest_framework lets add following. This includes following authentications globally.

```
REST_FRAMEWORK = {
                …
                  'DEFAULT_AUTHENTICATION_CLASSES': (
                                'rest_framework.authentication.BasicAuthentication',
                                 'rest_framework.authentication.TokenAuthentication',
                 )
}
```

Everytime you change settings.py run python manage.py migrate. This configures Django and applies the changes.

Go to your project/app/views.py and lets make the UserViewSet protected by authentication.

```
from rest_framework.views import APIView
from rest_framework.permissions import IsAuthenticated
```

Add to your UserVievSet class permission_classes = (IsAuthenticated,)

```
class UserViewSet(viewsets.ModelViewSet):
    """
    API endpoint that allows users to be viewed or edited.
    """
    permission_classes = (IsAuthenticated,)
    queryset = User.objects.all().order_by('-date_joined')
    serializer_class = UserSerializer
```

Now when you do GET request to your users view authentication is required. If connecting with browser a popup asking for username and password will show up. When valid authentication is not provided server will response with

"detail": "Authentication credentials were not provided."

Ref. 17

Ref. 18

8.1.7 Creating user and generating token

Now we have protected one endpoint with authentication but we cannot access it due to not having the token key. We go here how to generate token key manual way and in next section how to create token trough end api.

You can generate user through terminal.

$python manage.py createsuperuser –-username user1 --email user1@some.com

As for password lets add 1234 and bypass the password validation. Only when testing, if intended for actual use then recommend doing the password with care.

$python manage.py drf_create_token user1

Terminal should now return token key for your user. Response should look something like this

Generated token 1cf30b7f67929a8deb7073ca0e17fef1ad1bf64e for user user1

Now lets try accessing users page trough browser. Input your username and password. You should now be able to see all created users.

Now lets access trough postman. Set request type as GET and set url as http://public_ip.com:port/users/ open Headers tap. Create new key and name it as Authorization on value write Token <token_key>.

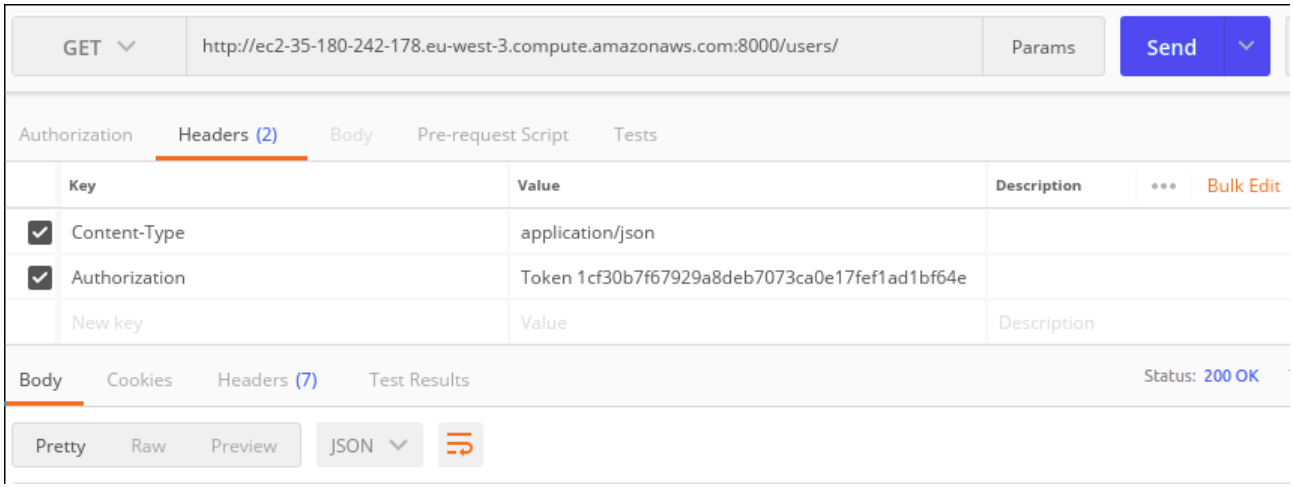


Illustration 21: Token authentication

Next we request token from already created user. Open your projects main urls.py.

Import rest_framework views.

```
from rest_framework.authtoken import views as authviews
```

We defined views as authviews due to having same named module from previous example. You cannot have two modules with same name in one module. Add under urlpatterns your endpoint.

```
urlpatterns = {
                ...
                path('auth/', authviews.obtain_auth_token, name = 'auth')
}
```

If you now send a GET request this will return {"detail":"Method \"GET\" not allowed."}

If you send a POST request with username and password token key will be returned.

This means users cannot simply open web browser and try accessing token keys, you need to provide valid username and password to get token key for provided user.

If we open Postmant and send a POST request to auth endpoint with following details. To URL set http://public_ip:port/auth/ and to your POST request body add

```
{
   "username": "your_user",
   "password": "your_user_password"
}
```

Once sended Django will return this users token key. If you have not created user, you can do this through terminal with manage.py.

You can generate user through terminal.

$python manage.py createsuperuser ––username your_user --email your_user@some.com

Django will then ask for password where you can simply set your_user_password and bypass the password validation.

Ref. 18

8.1.8 Creating user and generating token through end point

Next we are adding new end point to create and view users. When user is created this end point will automatically generate a token key.

Lets build a end point called create. We put this under

users/ → http://public_ip:port/users/create/

This end point will create new user to your database and return a token key.

Open your urls.py file project/urls.py and add there path to creating user

```
urlpatterns = [
                ....
                path('createUser', views.createUser),
]
```

Let’s add function to views.py to create user and token. This will grap the username and password from your POST body. Then validate it with database. If user found it will return token key if not found creates user and returns token key.

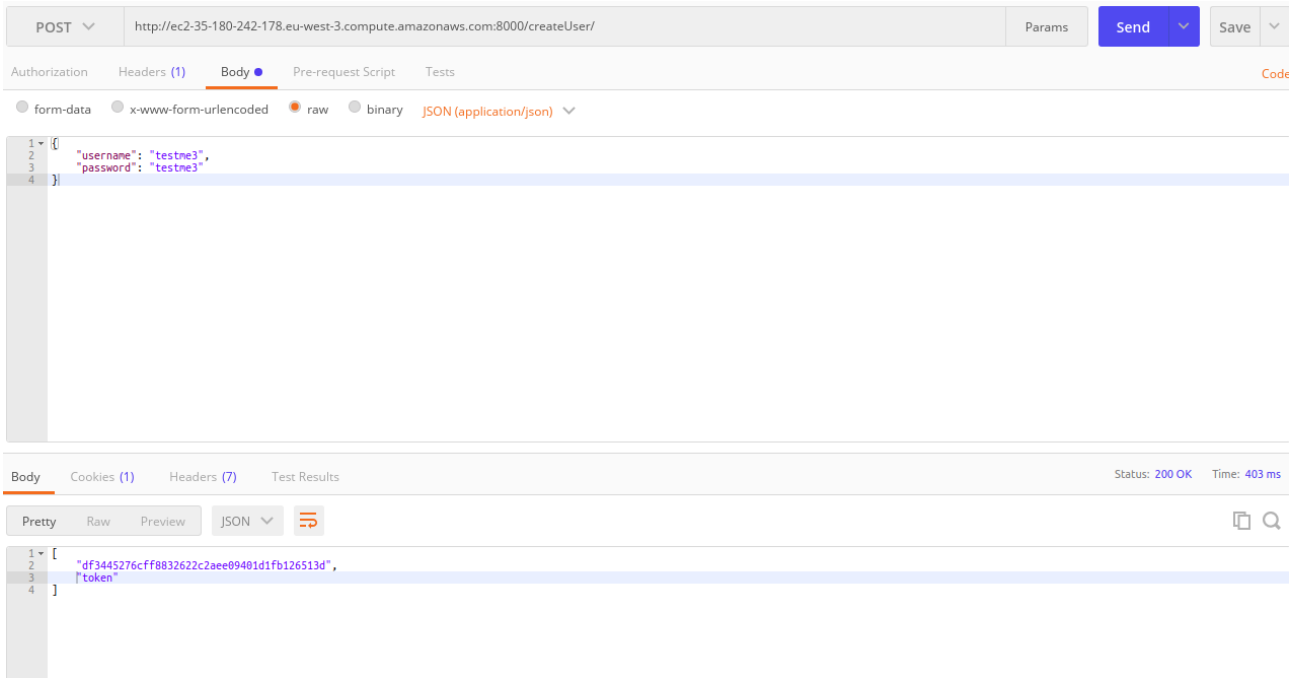
In case a variable is missing from the POST body, username or password. Bad request will be returned and information why request failed.

Open your apps views.py.

```
from rest_framework.status import (
    HTTP_400_BAD_REQUEST,
    HTTP_404_NOT_FOUND,
    HTTP_200_OK
)
from django.views.decorators.csrf import csrf_exempt
from rest_framework.decorators import api_view, permission_classes
from rest_framework.permissions import AllowAny
from django.contrib.auth import authenticate


@csrf_exempt
@api_view(["POST"])
@permission_classes((AllowAny,))
def createUser(request):
    username = request.data.get("username")
    password = request.data.get("password")

    if username is None or password is None:
        return Response({'error': 'Please provide both username and password'},
                status = HTTP_400_BAD_REQUEST)
```

```
user = authenticate(username = username, password = password)
if not user:
    user = User.objects.create_user(username=username, password = password)

token, _= Token.objects.get_or_create(user = user)

return Response({
    'token', token.key,}
    , status = HTTP_200_OK)
```

Try out with postman.



ref. 23

8.2 Install CUDA & CUDNN

Check that your GPU is capable for CUDA by typing to terminal

lspci | grep -i nvidia

Check the graphic drived model. I have GeForce GT 755M which is cuda enabled. Cuda enabled device table https://developer.nvidia.com/cuda-gpus

Verify you have gcc by typing

```
gcc --version
```

Download CUDA toolkit from https://developer.nvidia.com/cuda-downloads

and follow the installation instructions.

Also add PATH variable by typing in terminal

```
export PATH=/usr/local/cuda-10.0/bin${PATH:+:${PATH}}
```

To download CUDNN you have to create NVIDIA developer account. Download CUDNN from https://developer.nvidia.com/rdp/cudnn-download

Recommend to download .deb package if doing this on ubuntu.

Run following command in same folder as you downloaded this package

```
sudo dpkg -i libcudnn7_7.3.1.20-1+cuda10.0_amd64.deb
sudo dpkg -i libcudnn7-dev_7.0.3.11-1+cuda9.0_amd64.deb
```

Read instruction to install from https://docs.nvidia.com/deeplearning/sdk/cudnn-install/index.html

8.3 OpenCV 3.4.0 – Open source computer vision library

8.3.1 Install OpenCV on ubuntu

Make folder to your home directory and name it as Programming. Open folder and open terminal. Start by installing all necessary packages to support openCV.

We use here OpenCV 3.4.0 because Darknet Yolo dosen’t support newer versions at the moment.

Type following:

```
sudo apt-get install build-essential
sudo apt-get install cmake git
```

Essential packages for processing images:

```
sudo apt-get install libgtk2.0-dev pkg-config
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
```

Essential packages for processing videos:

```
sudo apt-get install libavcodec-dev libavformat-dev
```

sudo apt-get install libswscale-dev libv4l-dev

sudo apt-get install libxvidcore-dev libx264-dev

GUI:

sudo apt-get install libgtk-3-dev

sudo apt-get install libatlas-base-dev gfortran

For optimizing:

sudo apt-get install libatlas-base-dev

If want to use python, install following:

sudo apt-get install python2.7-dev python3.5-dev

Download and install OpenCV:

wget https://github.com/opencv/opencv/archive/3.4.0.zip -O opencv-3.4.0.zip

wget https://github.com/opencv/opencv_contrib/archive/3.4.0.zip -O opencv_contrib-3.4.0.zip

sudo apt-get install unzip

unzip opencv-3.4.0.zip

unzip opencv_contrib-3.4.0.zip

mv opencv_contrib-3.4.0.zip opencv-3.4.0/

Make a directory named build inside OpenCV-3.4.0:

cd opencv-3.4.0

mkdir build

cd build

Compile opencv:

cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=/usr/local
-DOPENCV_EXTRA_MODULES_PATH=../opencv_contrib-3.4.0/modules
-DOPENCV_ENABLE_NONFREE=True ..

Make openCV, j7 means use 7 cores for building:

make -j7

sudo make install

sudo ldconfig

Congraz you are done.

8.4 Object detection with Darknet YOLO

8.4.1 Install darknet

Darknet Yolo is a real time object detection software. Very usefull for detecting large object and for fast processing speed. Ref https://github.com/AlexeyAB/darknet

If nvcc not found error occurs and you have installed CUDA → type following to terminal export PATH=/usr/local/cuda-10.0/bin${PATH:+:${PATH}}

Download darknet.

- Get git link for darknet from https://github.com/AlexeyAB/darknet
- In folder where you want to install darknet type to terminal ”git clone https://github.com/AlexeyAB/darknet.git”

Install darknet with software processing

- Open cloned folder and type in terminal ”make”

Install darknet with GPU processing. CUDA 9.2(TODO link for example here) is required

- Open cloned folder and edit Makefile by typing ”nano Makefile”
- Enable GPU processing by changing GPU=0 to GPU=1.
- Enable CUDNN by changing 0 to 1
- Save and exit editor.
- Type ”make” to terminal

Install darknet with GPU processing and OpenCV(TODO how to install OpenCV)

- Open cloned folder and edit Makefile by typing ”nano Makefile”
- Enable GPU processing by changing GPU=0 to GPU=1.
- Enable CUDNN by changing 0 to 1
- Enable OpenCV by changing OPENCV=0 to OPENCV=1

- Type ”make” to terminal

Download pre-trained data file.

- Type in terminal ”wget https://pjreddie.com/media/files/yolov3.weights”.

If no error occured during building process you are done :).

8.4.2 Running YOLO and detecting objects

Go into darknet directory type to terminal

”./darknet detector test cfg/coco.data cfg/yolov3.cfg yolov3.weights”. After loading trained data terminal will prompt text ”Enter Image Path:” type here ”data/eagle.jpg”. Window displaying prediction should popout. Prediction should show blue rectangle around eagle and show prediction reliability in terminal ”bird: 99%”

8.4.3 Training YOLO to recognize custom objects

In this example we make yolo to detect karhu branded beer. Neural network is given 3 objects to detect. Karhu in bottle, Karhu in can and a real karhu. Karhu is finnish and means bear.

1. Go to darknet folder
2. Open cfg folder
3. Copy yolov3.cfg to yolo-karhu-obj.cfg → cp yolov3.cfg yolo-karhu-obj.cfg
4. Open yolo-karhu-obj.cfg in editor → gedit yolo-karhu-obj.cfg
5. Modify batch to 64 and subdivisions to 8
    1. batch=64
    2. subdivision=8
6. Change line classes=80. Classes represents amount of objects detected. In our case we have 3 objects.
    1. Karhu in a can
    2. Karhu in a bottle
    3. A real karhu

    There are 3 lines for classes and 3 lines for filters to change. Set filter value with this equation . filters=(classes + 5)x3. In our case we have 24 filters.

    4. Change classes to 3 at line 610, line 696, line 783
    5. Change filters to 24 at line 603, 689, 776
7. Open directory build\darknet\x64\data
8. Create file karhu-obj.names and add the object names. → file should look like follow.

Karhu in can

Karhu in bottle

A real karhu

9. Create file karhu-obj.data and add follow

   classses=3

   train = data/karhu-train.txt

   valid = data/test.txt

   names = karhu-obj.names

   backup = backup/

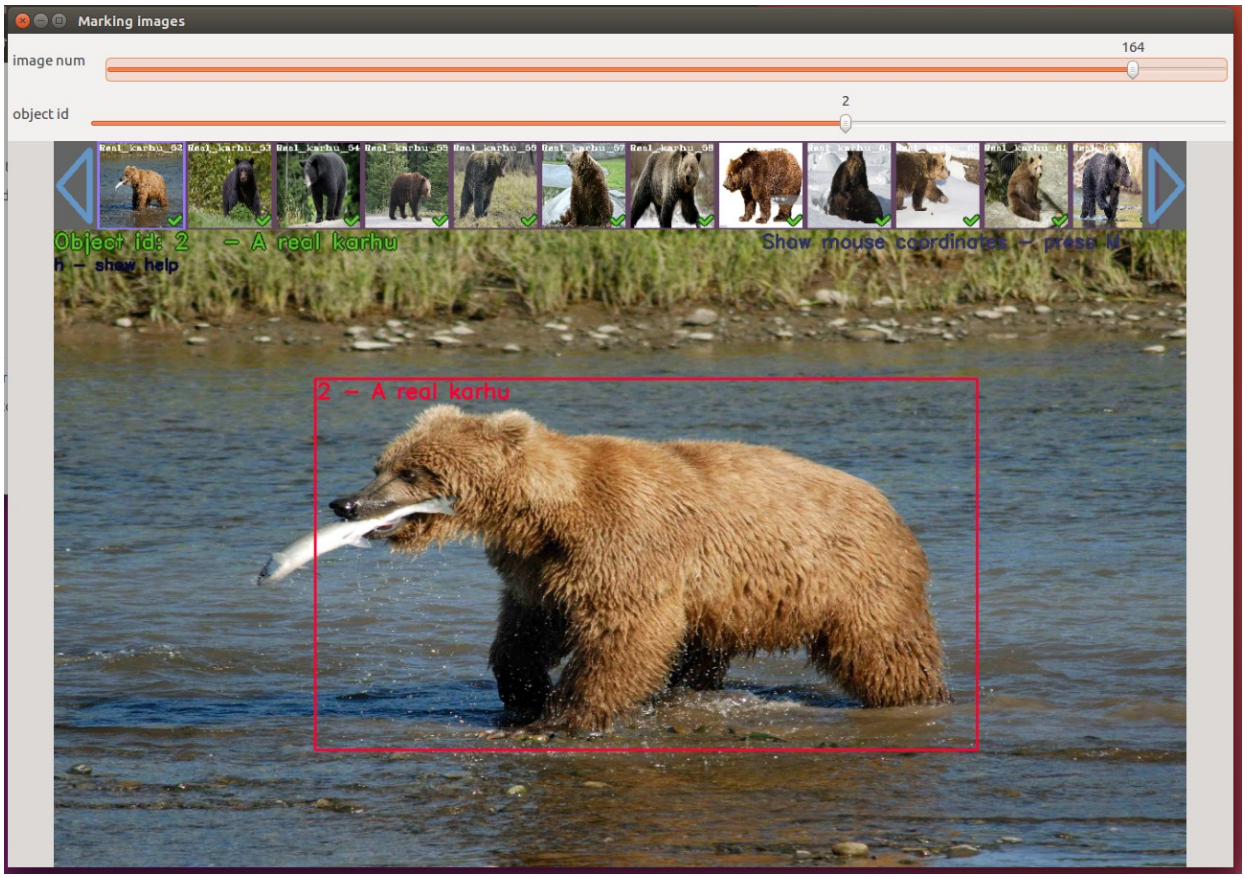
10. Create folder obj and add all images to be trained here. You can also download a zip package containing training images from https://drive.google.com/open?id=16l4ZDc_82_LIQqxuxCH2uyTBcS3omz0u
11. Return back to darknet folder
12. Download GUI for marking bounding objects from https://github.com/AlexeyAB/Yolo_mark
13. clone to darknet folder → git clone https://github.com/AlexeyAB/Yolo_mark.git
14. Enter cloned folder Yolo_mark and compile
15. Run cmake . && make
16. Go back to *build*/darknet/x64/data and create a python file → touch ”generateImageList.py”
17. Copy pase following into generateImageList.py

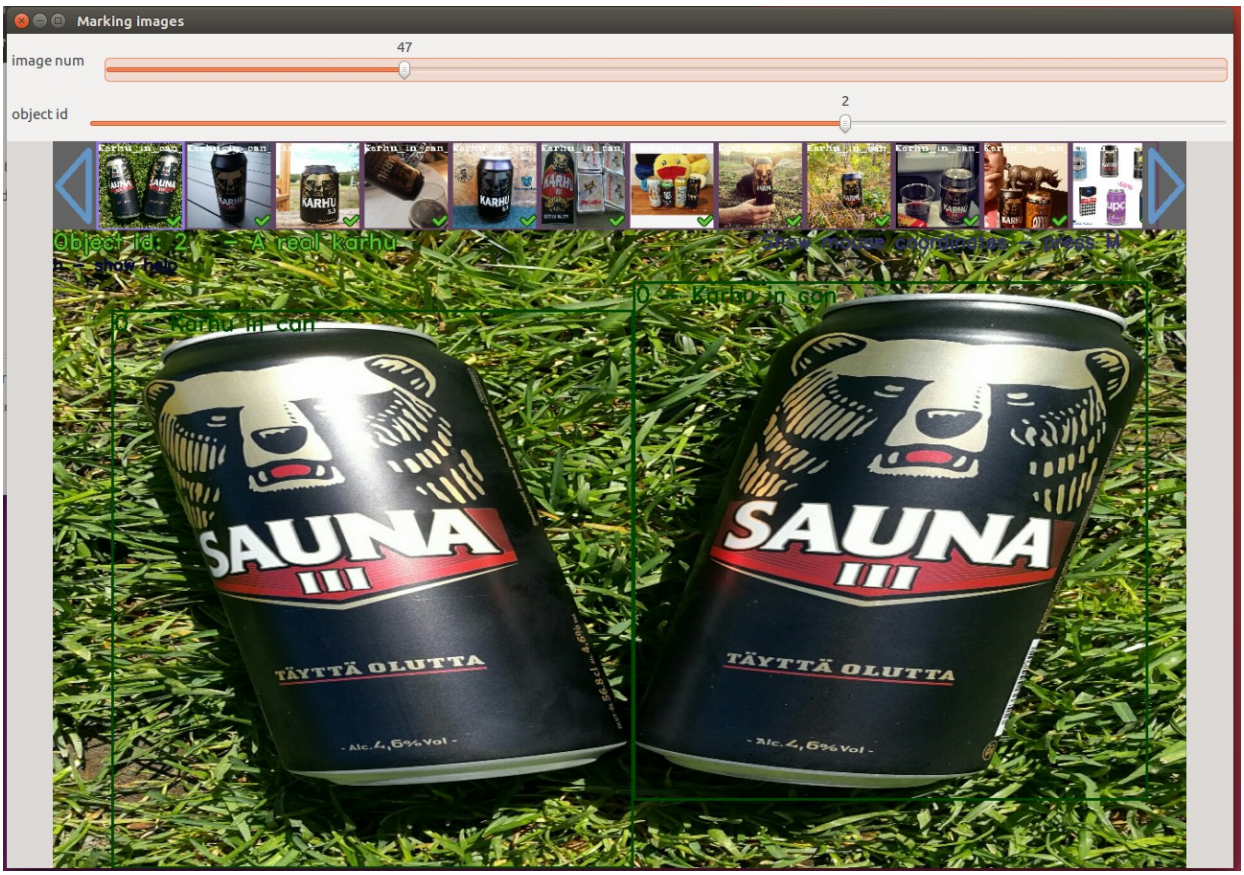
```
#This software generates imagelist based on /obj/ folder
import os
file = open("karhu-train.txt", "w") # Create train.txt, will overwrite previous
for root, dirs, files in os.walk("obj/"): # Search all files from obj/folder
  for filename in files:
    if ".txt" in filename:
      continue
    filename = "build/darknet/x64/data/obj/" + filename
    file.write(filename + "\n\r") # Write filepath to train.txt file
    print(filename)# Print filepath to terminal
file.close()  # Close file
```

18. Download pre-trained weights https://pjreddie.com/media/files/darknet53.conv.74

Extract to build\darknet\x64. Quick way to do this is typing in the desired folder wget https://pjreddie.com/media/files/darknet53.conv.74. This downloads the file straight to working directory.

19. Goto folder where Yolo-Mark was installed and run the software ./yolo_mark ../build/darknet/x64/data/obj ../build/darknet/x64/data/karhu-train.txt ../build/darknet/x64/data/karhu-obj.names





20. Exit yolo mark folder and start training by running darknet

    ./darknet detector train build/darknet/x64/data/karhu-obj.data build/darknet/x64/cfg/yolo-karhu-obj.cfg build/darknet/x64/darknet53.conv.74 -dont_show

21. If darknet gets stuck or gets killed while loading shortcuts layer. Change subdivisions in .cfg file to 16.

    If darknet gets stuck while loading image txt file, check that this txt file doesn’t have any 0 coordinates there. This causes crash, change 0 to 0.1. Numbers should be between 0 – 1.

    If error cuda out of memory occurs while running training, run it other terminal nvidia-smi and monitor GPU memory.

22. Training darknet is going to take awfully lot of time. Training on jetson around 8h will get you 400 -500 iterations. At least 10 000 iterations are required.

    After 10 000 iteration darknet will build the weight file first time.

23. Under progress, Have not reached this yet. Waiting for training to finish :).

9 Investment & Financing

Email template for Ry contacting third party regarding funding, advices and meeting.

Email Structure

Greetings,

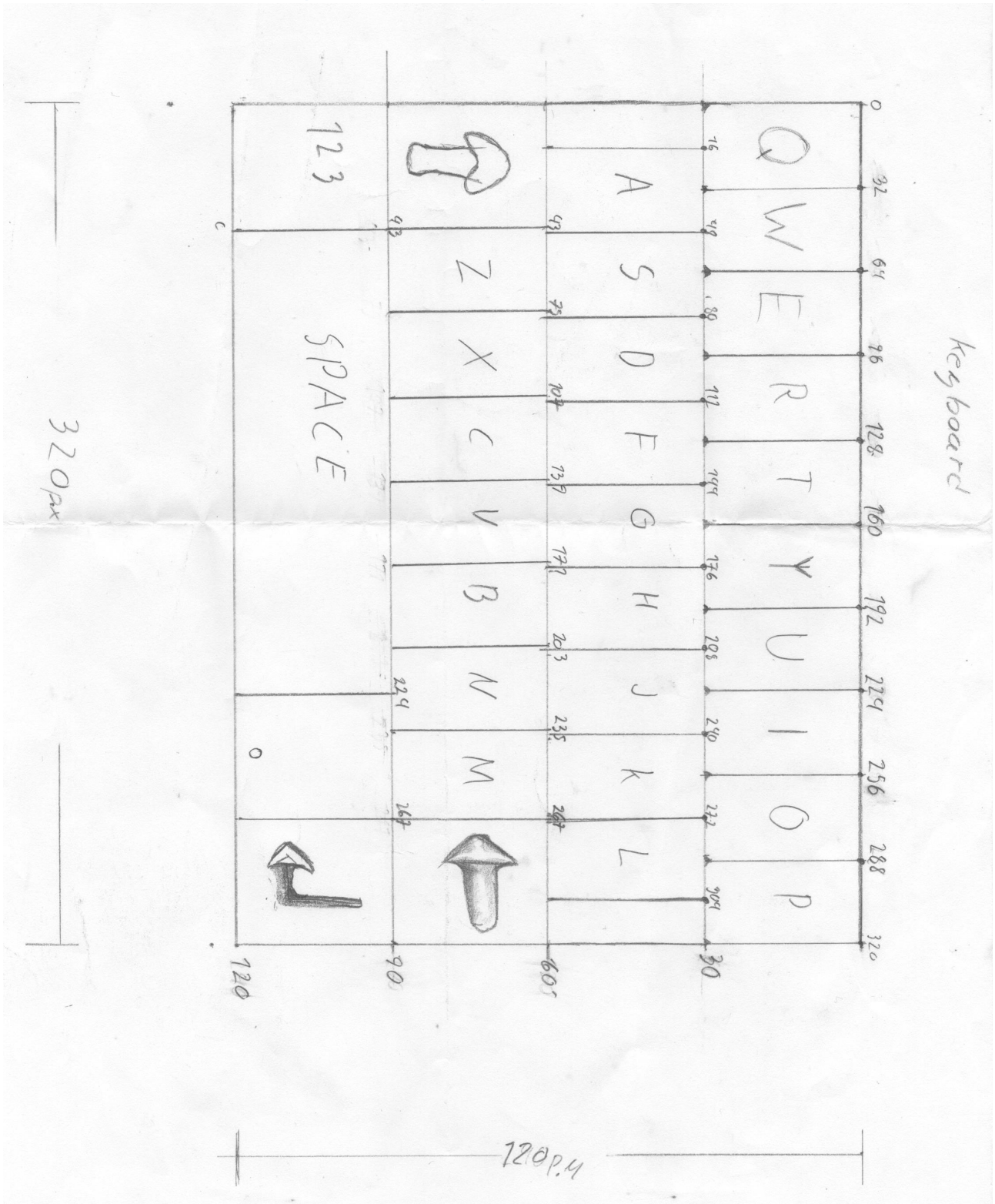
Why we contact them.
What are our goals and what we do.
How we going to achieve this.
Repetition of why we contacted and what we looking for. Request to meet.
Explanation where we stand, size of community, where to get more information of us.
What are the fundings and guidances requested for.

Goodbyes and signature

---

10 Examples #TODO move these to seperate files

10.1 Keyboard example code



When ok is pushed program will print inputted string to computer (Teraterm/Putty)

```
#include "mbed.h"
#include "SPI_TFT_ILI9341.h"
#include "Arial12x12.h"
#include "Arial24x23.h"
#include "Arial28x28.h"
#include "font_big.h"
#include "SeeedStudioTFTv2.h"

SeeedStudioTFTv2 tft(p19,p18,p17,p16,p11, p12, p13, p14, p22, p20);
Serial pc(USBTX, USBRX);

// Keyboard
char key;   //
char chatx; //
char x2;    //
char chaty; //
char key_i;
char key_back[120];
char t, t2, t3, t4, t5, t6;
char key_skip;
char buffer[100];

PwmOut Brightness(p25);// brightness
DigitalOut myled(LED1);

point p;
void touch()
 {
   while(1)
    {
       p.x=0;p.y=0;
       if (tft.getTouch(p)==tft.YES) // read analog pos.
```

```
      {
        tft.getTouch(p);
        tft.getPixel(p);            // convert to pixel pos
        pcui.printf("\n\r Point x:%d, y:%d", p.x, p.y);

       wait_ms(150);
        break;
      }
    }
 }
int main() {

// brightness control
Brightness = 1;  // Set brightness control on
Brightness.period(0.005);

// Prepare screen
tft.claim(stdout);      // send stdout to the TFT display
tft.set_orientation(1); // Direction of printing
tft.cls();  // Clear screen
tft.set_font((unsigned char*) Arial12x12);  // Set font size
tft.foreground(White);  // Set foreground color as White
tft.background(Black);  // Set background color as Black
tft.locate(70,120); // Locate start pixel area. This is where printf command starts printing.
// First characters top left is now 70,120
tft.printf("Calibrating touch screen");
wait(1);

tft.cls();

tft.calibrate();
```

```
while(1) {

   // Text Box

   tft.fillrect(0,0,250,120,LightGrey);

   // Reset buffer

   for(int i=0; i<100; i++) {buffer[i]=' ';}

   chatx=1;
   x2=0;
   chaty=1;
   t=1;t3=0;t4=0;t5=0;t6=0;
   t2=0;
   key_skip=0;
   key_i=1;

   // general
   skip2:
   tft.set_orientation(1);
   tft.set_font((unsigned char*) Arial12x12);
   tft.foreground(White);
   tft.background(Black);

      tft.fillrect(0,120,320,240,Black);
      tft.fillrect(250,0,320,120,Black);

 // Keyboard Frame
   tft.line(0,120,320,120,Red); tft.line(0,150,320,150,Red); tft.line(0,180,320,180,Red);
tft.line(0,210,320,210,Red);
```

```
   tft.line(0,240,320,240,Red); tft.line(250,0,250,120, Red);

      // Inlines

   tft.line(32,120,32,150,Red); tft.line(64,120,64,150,Red); tft.line(96,120,96,150,Red);
tft.line(128,120,128,150,Red);

   tft.line(160,120,160,150,Red); tft.line(192,120,192,150,Red);
tft.line(224,120,224,150,Red); tft.line(256,120,256,150,Red);

   tft.line(288,120,288,150,Red);

      // Keyboard second row, inlines

   tft.line(16,150,16,180,Red); tft.line(48,150,48,180,Red); tft.line(80,150,80,180,Red);
tft.line(112,150,112,180,Red);

   tft.line(144,150,144,180,Red); tft.line(176,150,176,180,Red);
tft.line(208,150,208,180,Red); tft.line(240,150,240,180,Red);

   tft.line(272,150,272,180,Red); tft.line(304,150,304,180,Red);

      // Keyboard third row, inlines

   tft.line(43,180,43,210,Red);

tft.line(75,180,75,210,Red);

tft.line(107,180,107,210,Red); tft.line(139,180,139,210,Red);

   tft.line(171,180,171,210,Red); tft.line(203,180,203,210,Red);
tft.line(235,180,235,210,Red); tft.line(267,180,267,210,Red);

    // Keyboard fourth row, inlines

   tft.line(43,210,43,240,Red); tft.line(224,210,224,240,Red);
tft.line(267,210,267,240,Red);


    switch(t)

    {

    // Lowercase

    case 1:


   // Keyboard first row


   // Butt3ons

   tft.locate(12,130);  tft.putc('q');

   tft.locate(45,130);  tft.putc('w');

   tft.locate(77,130);  tft.putc('e');
```

```
tft.locate(109,130); tft.putc('r');
tft.locate(141,130); tft.putc('t');
tft.locate(174,130); tft.putc('y');
tft.locate(205,130); tft.putc('u');
tft.locate(237,130); tft.putc('i');
tft.locate(269,130); tft.putc('o');
tft.locate(301,130); tft.putc('p');



    // Butt3ons
tft.locate(28,160);  tft.putc('a');
tft.locate(60,160);  tft.putc('s');
tft.locate(92,160);  tft.putc('d');
tft.locate(124,160); tft.putc('f');
tft.locate(156,160); tft.putc('g');
tft.locate(188,160); tft.putc('h');
tft.locate(220,160); tft.putc('j');
tft.locate(252,160); tft.putc('k');
tft.locate(284,160); tft.putc('l');

        // Butt3ons
tft.locate(10,190);  tft.printf("A/a");
tft.locate(55,190);  tft.putc('z');
tft.locate(87,190);  tft.putc('x');
tft.locate(118,190); tft.putc('c');
tft.locate(151,190); tft.putc('v');
tft.locate(183,190); tft.putc('b');
tft.locate(215,190); tft.putc('n');
tft.locate(247,190); tft.putc('m');
tft.set_font((unsigned char*) Arial24x23);
tft.locate(273,186); tft.printf("<~");
```

```
// Butt3ons
tft.set_font((unsigned char*) Arial12x12);
tft.locate(6,220);   tft.printf("123");
tft.set_font((unsigned char*) Arial24x23);
tft.locate(80,215);  tft.printf("SPACE");
tft.locate(238,215); tft.putc('.');
tft.locate(274,215); tft.printf("<]");
tft.locate(263,50); tft.printf("OK");

   break;
 //Uppercase
 case 2:

       // Butt3ons
        tft.locate(12,130);  tft.putc('Q');
        tft.locate(45,130);  tft.putc('W');
        tft.locate(77,130);  tft.putc('E');
        tft.locate(109,130); tft.putc('R');
        tft.locate(141,130); tft.putc('T');
        tft.locate(174,130); tft.putc('Y');
        tft.locate(205,130); tft.putc('U');
        tft.locate(237,130); tft.putc('I');
        tft.locate(269,130); tft.putc('O');
        tft.locate(301,130); tft.putc('P');

       // Buttons
        tft.locate(28,160);  tft.putc('A');
        tft.locate(60,160);  tft.putc('S');
        tft.locate(92,160);  tft.putc('D');
        tft.locate(124,160); tft.putc('F');
        tft.locate(156,160); tft.putc('G');
        tft.locate(188,160); tft.putc('H');
```

```
        tft.locate(220,160); tft.putc('J');
        tft.locate(252,160); tft.putc('K');
        tft.locate(284,160); tft.putc('L');

       // Buttons
        tft.locate(10,190);  tft.printf("A/a");
        tft.locate(55,190);  tft.putc('Z');
        tft.locate(87,190);  tft.putc('X');
        tft.locate(118,190); tft.putc('C');
        tft.locate(151,190); tft.putc('V');
        tft.locate(183,190); tft.putc('B');
        tft.locate(215,190); tft.putc('N');
        tft.locate(247,190); tft.putc('M');
        tft.set_font((unsigned char*) Arial24x23);
        tft.locate(273,186); tft.printf("<~");


       // Buttons
        tft.set_font((unsigned char*) Arial12x12);
        tft.locate(6,220);   tft.printf("123");
        tft.set_font((unsigned char*) Arial24x23);
        tft.locate(80,215);  tft.printf("SPACE");
        tft.locate(238,215); tft.putc('.');
        tft.locate(274,215); tft.printf("<]");
        tft.locate(263,50); tft.printf("OK");

       break;

  // Numbers
  case 3:

  // general
```

```
tft.set_font((unsigned char*) Arial12x12);

// Keyboard first row

// Buttons
tft.locate(12,130);  tft.putc('1');
tft.locate(45,130);  tft.putc('2');
tft.locate(77,130);  tft.putc('3');
tft.locate(109,130); tft.putc('4');
tft.locate(141,130); tft.putc('5');
tft.locate(174,130); tft.putc('6');
tft.locate(205,130); tft.putc('7');
tft.locate(237,130); tft.putc('8');
tft.locate(269,130); tft.putc('9');
tft.locate(301,130); tft.putc('0');

// Keyboard second row, inlines

    // Buttons
tft.locate(28,160);  tft.putc('!');
tft.locate(60,160);  tft.putc('#');
tft.locate(92,160);  tft.putc('%');
tft.locate(124,160); tft.putc('&');
tft.locate(156,160); tft.putc('/');
tft.locate(188,160); tft.putc('(');
tft.locate(220,160); tft.putc(')');
tft.locate(252,160); tft.putc('=');
tft.locate(284,160); tft.putc('?');

// Keyboard third row, inlines

        // Buttons
```

```
  tft.locate(10,190);  tft.printf("A/a");
  tft.locate(55,190);  tft.putc('<');
  tft.locate(87,190);  tft.putc('>');
  tft.locate(118,190); tft.putc(',');
  tft.locate(151,190); tft.putc(';');
  tft.locate(183,190); tft.putc(':');
  tft.locate(215,190); tft.putc('-');
  tft.locate(247,190); tft.putc('_');
  tft.set_font((unsigned char*) Arial24x23);
  tft.locate(273,186); tft.printf("<~");


  // Buttons
  tft.set_font((unsigned char*) Arial12x12);
  tft.locate(6,220);   tft.printf("123");
  tft.set_font((unsigned char*) Arial24x23);
  tft.locate(80,215);  tft.printf("SPACE");
  tft.locate(238,215); tft.putc('.');
  tft.locate(274,215); tft.printf("<]");
  tft.locate(263,50); tft.printf("OK");

  // touch effect

      tft.background(LightGrey);



     break;
   };

  while(1)
  {
     touch();
```

```
      // Keyboard first row
      switch(t)
      {
      case 2:
       // Debug
       // pc.printf("\n\r Keyboard Case 2. P.x:%d and P.y:%d",p.x,p.y);
       tft.background(LightGrey);

          if((p.x>0 && p.x<=31) && (p.y>=120 && p.y<=149))   {key='Q';};  // Q
          if((p.x>=32 && p.x<=63) && (p.y>=120 && p.y<=149))  {key='W';};  // W
          if((p.x>=64 && p.x<=97) && (p.y>=120 && p.y<=149))  {key='E';};  // E
          if((p.x>=98 && p.x<=127) && (p.y>=120 && p.y<=149)) {key='R';};  // R
          if((p.x>=128 && p.x<=159) && (p.y>=120 && p.y<=149)){key='T';};  // T
          if((p.x>=160 && p.x<=191) && (p.y>=120 && p.y<=149)){key='Y';};  // Y
          if((p.x>=192 && p.x<=223) && (p.y>=120 && p.y<=149)){key='U';};  // U
          if((p.x>=224 && p.x<=255) && (p.y>=120 && p.y<=149)){key='I';};  // I
          if((p.x>=256 && p.x<=287) && (p.y>=120 && p.y<=149)){key='O';};  // O
          if((p.x>=288 && p.x<=320) && (p.y>=120 && p.y<=149)){key='P';};  // P
          // Keyboard second row
          if((p.x>=16 && p.x<=47) && (p.y>=150 && p.y<=179))  {key='A';};  // A
          if((p.x>=48 && p.x<=79) && (p.y>=150 && p.y<=179))  {key='S';};  // S
          if((p.x>=80 && p.x<=111) && (p.y>=150 && p.y<=179)) {key='D';};  // D
          if((p.x>=112 && p.x<=143) && (p.y>=150 && p.y<=179)){key='F';};  // F
          if((p.x>=144 && p.x<=175) && (p.y>=150 && p.y<=179)){key='G';};  // G
          if((p.x>=176 && p.x<=207) && (p.y>=150 && p.y<=179)){key='H';};  // H
          if((p.x>=208 && p.x<=239) && (p.y>=150 && p.y<=179)){key='J';};  // J
          if((p.x>=240 && p.x<=271) && (p.y>=150 && p.y<=179)){key='K';};  // K
          if((p.x>=272 && p.x<=303) && (p.y>=150 && p.y<=179)){key='L';};  // L
          // Keyboard third row
          if((p.x>0 && p.x<=42) && (p.y>=180 && p.y<=209)){t=1; goto skip2;};     //
uppercase/lowercase
```

```
          if((p.x>=43 && p.x<=73) && (p.y>=180 && p.y<=209)){key='Z';};    // Z
          if((p.x>=75 && p.x<=106) && (p.y>=180 && p.y<=209)){key='X';};   // X
          if((p.x>=107 && p.x<=138) && (p.y>=180 && p.y<=209)){key='C';};  // C
          if((p.x>=139 && p.x<=170) && (p.y>=180 && p.y<=209)){key='V';};  // V
          if((p.x>=171 && p.x<=202) && (p.y>=180 && p.y<=209)){key='B';};  // B
          if((p.x>=203 && p.x<=234) && (p.y>=180 && p.y<=209)){key='N';};  // N
          if((p.x>=235 && p.x<=266) && (p.y>=180 && p.y<=209)){key='M';};  // M
          if((p.x>=267 && p.x<=320) && (p.y>=180 && p.y<=209)){key='\b';}; //
backspace


          // Keyboard fourth row
          if((p.x>0 && p.x<=42) && (p.y>=210 && p.y<=240)){t=3; goto
skip2;};        // 123
          if((p.x>=43 && p.x<=223) && (p.y>=210 && p.y<=240)){key=' ';};    //
Space
          if((p.x>=224 && p.x<=266) && (p.y>=210 && p.y<=240)){key='.';};   // dot
          if((p.x>=267 && p.x<=320) && (p.y>=210 && p.y<=240)){key='\n';};  //
enter
          break;


       case 1:


         //Debug
       // pc.printf("\n\r Keyboard Case 1. P.x:%d and P.y:%d",p.x,p.y);
        tft.background(LightGrey);
          if((p.x>0 && p.x<=31) && (p.y>=120 && p.y<=149))   {key='q';};     // q
          if((p.x>=32 && p.x<=63) && (p.y>=120 && p.y<=149))  {key='w';};  // w
          if((p.x>=64 && p.x<=97) && (p.y>=120 && p.y<=149))  {key='e';};  // e
          if((p.x>=98 && p.x<=127) && (p.y>=120 && p.y<=149)) {key='r';};  // r
          if((p.x>=128 && p.x<=159) && (p.y>=120 && p.y<=149)){key='t';};  // t
          if((p.x>=160 && p.x<=191) && (p.y>=120 && p.y<=149)){key='y';};  // y
          if((p.x>=192 && p.x<=223) && (p.y>=120 && p.y<=149)){key='u';};  // u
          if((p.x>=224 && p.x<=255) && (p.y>=120 && p.y<=149)){key='i';};  // i
          if((p.x>=256 && p.x<=287) && (p.y>=120 && p.y<=149)){key='o';};  // o
```

```
        if((p.x>=288 && p.x<=320) && (p.y>=120 && p.y<=149)){key='p';};  // p
        // Keyboard second row
        if((p.x>=15 && p.x<48) && (p.y>=150 && p.y<=179))  {key='a';};  // a
        if((p.x>=48 && p.x<80) && (p.y>=150 && p.y<=179))  {key='s';};  // s
        if((p.x>=80 && p.x<111) && (p.y>=150 && p.y<=179)) {key='d';};  // d
        if((p.x>=112 && p.x<144) && (p.y>=150 && p.y<=179)){key='f';};  // f
        if((p.x>=144 && p.x<175) && (p.y>=150 && p.y<=179)){key='g';};  // g
        if((p.x>=176 && p.x<208) && (p.y>=150 && p.y<=179)){key='h';};  // h
        if((p.x>=208 && p.x<240) && (p.y>=150 && p.y<=179)){key='j';};  // j
        if((p.x>=240 && p.x<272) && (p.y>=150 && p.y<=179)){key='k';};  // k
        if((p.x>=272 && p.x<304) && (p.y>=150 && p.y<=179)){key='l';};  // l
        // Keyboard third row
        if((p.x>=1 && p.x<=42) && (p.y>=180 && p.y<=209)){t=2; goto
skip2;};     // uppercase/lowercase
        if((p.x>=43 && p.x<=73) && (p.y>=180 && p.y<=209)){key='z';};    // z
        if((p.x>=74 && p.x<=106) && (p.y>=180 && p.y<=209)){key='x';};   // x
        if((p.x>=107 && p.x<=138) && (p.y>=180 && p.y<=209)){key='c';};  // c
        if((p.x>=139 && p.x<=170) && (p.y>=180 && p.y<=209)){key='v';};  // v
        if((p.x>=171 && p.x<=202) && (p.y>=180 && p.y<=209)){key='b';};  // b
        if((p.x>=203 && p.x<=234) && (p.y>=180 && p.y<=209)){key='n';};  // n
        if((p.x>=235 && p.x<=266) && (p.y>=180 && p.y<=209)){key='m';};  // m
        if((p.x>=267 && p.x<=320) && (p.y>=180 && p.y<=209)){key='\b';}; //
backspace


        // Keyboard fourth row
        if((p.x>0 && p.x<=42) && (p.y>=210 && p.y<=240)){t=3; goto
skip2;};       // 123
        if((p.x>=43 && p.x<=223) && (p.y>=210 && p.y<=240)){key=' ';};    //
Space
        if((p.x>=224 && p.x<=266) && (p.y>=210 && p.y<=240)){key='.';};   // dot
        if((p.x>=267 && p.x<=320) && (p.y>=210 && p.y<=240)){key='\n';};  //
enter
      break;
```

```
       // Numbers
       case 3:


      // Debug
    // pc.printf("\n\r Keyboard Case 3. P.x:%d and P.y:%d",p.x,p.y);
        tft.background(LightGrey);
          if((p.x>0 && p.x<=31) && (p.y>=120 && p.y<=149))   {key='1';};     // 1
          if((p.x>=32 && p.x<=63) && (p.y>=120 && p.y<=149))  {key='2';};  // 2
          if((p.x>=64 && p.x<=97) && (p.y>=120 && p.y<=149))  {key='3';};  // 3
          if((p.x>=98 && p.x<=127) && (p.y>=120 && p.y<=149)) {key='4';};  // 4
          if((p.x>=128 && p.x<=159) && (p.y>=120 && p.y<=149)){key='5';};  // 5
          if((p.x>=160 && p.x<=191) && (p.y>=120 && p.y<=149)){key='6';};  // 6
          if((p.x>=192 && p.x<=223) && (p.y>=120 && p.y<=149)){key='7';};  // 7
          if((p.x>=224 && p.x<=255) && (p.y>=120 && p.y<=149)){key='8';};  // 8
          if((p.x>=256 && p.x<=287) && (p.y>=120 && p.y<=149)){key='9';};  // 9
          if((p.x>=288 && p.x<=320) && (p.y>=120 && p.y<=149)){key='0';};  // 0
          // Keyboard second row
          if((p.x>=16 && p.x<=47) && (p.y>=150 && p.y<=179))  {key='!';};  // !
          if((p.x>=48 && p.x<=79) && (p.y>=150 && p.y<=179))  {key='#';};  // #
          if((p.x>=80 && p.x<=111) && (p.y>=150 && p.y<=179)) {key='%';};  // %
          if((p.x>=112 && p.x<=143) && (p.y>=150 && p.y<=179)){key='&';};  // &
          if((p.x>=144 && p.x<=175) && (p.y>=150 && p.y<=179)){key='/';};  // /
          if((p.x>=176 && p.x<=207) && (p.y>=150 && p.y<=179)){key='(';};  // (
          if((p.x>=208 && p.x<=239) && (p.y>=150 && p.y<=179)){key=')';};  // )
          if((p.x>=240 && p.x<=271) && (p.y>=150 && p.y<=179)){key='=';};  // =
          if((p.x>=272 && p.x<=303) && (p.y>=150 && p.y<=179)){key='?';};  // ?
          // Keyboard third row
          if((p.x>0 && p.x<=42) && (p.y>=180 && p.y<=209)){t=1; goto skip2;};     //
uppercase/lowercase
          if((p.x>=41 && p.x<=73) && (p.y>=180 && p.y<=209)){key='<';};    // <
          if((p.x>=74 && p.x<=106) && (p.y>=180 && p.y<=209)){key='>';};   // >
          if((p.x>=107 && p.x<=138) && (p.y>=180 && p.y<=209)){key=',';};  // ,
```

```
          if((p.x>=139 && p.x<=170) && (p.y>=180 && p.y<=209)){key=';';};  // ;
          if((p.x>=170 && p.x<=202) && (p.y>=180 && p.y<=209)){key=':';};  // :
          if((p.x>=203 && p.x<=234) && (p.y>=180 && p.y<=209)){key='-';};  // -
          if((p.x>=235 && p.x<=266) && (p.y>=180 && p.y<=209)){key='_';};  // _
          if((p.x>=267 && p.x<=320) && (p.y>=180 && p.y<=209)){key='\b';}; //
Backspace


          // Keyboard fourth row
          if((p.x>0 && p.x<=42) && (p.y>=210 && p.y<=240)){t=3; goto
skip2;};        // 123
          if((p.x>=43 && p.x<=223) && (p.y>=210 && p.y<=240)){key=' ';};    //
Space
          if((p.x>=224 && p.x<=266) && (p.y>=210 && p.y<=240)){key='.';};   // Dot
          if((p.x>=267 && p.x<=320) && (p.y>=210 && p.y<=240)){key='\n';};  //
Enter
       break;
       };

/**********************************************************************************
*************/
       // Chat Box engine
      if((p.x>250 && p.x<=320) && (p.y>0 && p.y<=119)){goto key_break;}    // OK is
pressed
      if((p.x>0 && p.x<=320) && (p.y>120 && p.y<=240)) // If key board is pressed
      {
       // if((key=='\b') && (chatx>=5) && (x2>0) && (t==1 || t==3)){x2--; key=' ';
chatx=(chatx-20);t2=1;}
       // if((key=='\b') && (chatx>=5) && (x2>0) && (t==2)){x2--; key=' ';
chatx=(chatx-22);t2=1;}
       if(key=='\b')
       {
          if((key_back[key_i]=='\n') && (chaty>23)){chaty=(chaty-23); t5=1;} // print
          t2=1;t4=1;
          if(key_i<1){key_i=1; key_back[key_i]=23;}
          key=' ';
```

```
          if(t3!=3){t3=3;key_i--;}
          //if(x2<1){buffer[0]=',';x2=1;};
          if(x2>0){x2--;}
       }
       if(t2!=1 && t4==1){key_i++;t4=0;}
       if((key=='\n') && (chaty<80)){
          chaty=(chaty+23);
          key_back[key_i]='\n';
          key_i++; key_back[key_i]=1;}

      if(key!='\n')
      {

        if(t2==1){goto skip;}
        if((key!='\b') && ((p.x>0) && (p.x<=320)) && ((p.y>=120) && (p.y<=240)))
          {

             if(t==1 || t==3) { key_back[key_i]=(key_back[key_i-1]+20);}
             if(t==2) { key_back[key_i]=(key_back[key_i-1]+22);}
             if((key_back[key_i]>230) && (chaty<80)){chaty=(chaty+23);
key_back[key_i]='\n';key_i++; key_back[key_i]=23;}
          }
       }

       skip:
       buffer[x2]=key;
       if(chaty>80 && key_back[key_i]>=200 && t2!=1)
       {
          key='*';

          if(t==1 || t==3) { key_back[key_i]=(key_back[key_i-1]-20);}
          if(t==2) { key_back[key_i]=(key_back[key_i-1]-22);}
```

```
        key_i--;
      }
      if(key_back[key_i]<200){key_skip=0;}

      if(t5!=1){
        tft.locate(key_back[key_i],chaty);
        tft.putc(key);

        if(key!='*' && key!='\n') { buffer[x2]=key; }
        if(key=='\n') { buffer[x2]=' '; }
        // Debug
        // pc.printf("\n\r Buffer: %s", buffer);
       } // print
      t5=0;

      //ADDS & SUBS
      if(t2==1)
        {
         key_i--;
        }

      if(t2!=1){
        key_i++;
        t3=0;
        if(key!='*') { x2++; }
        }
      t2=0;
      wait(0.1);
      };

/*************************************************************************************
*********************/
    }
```

```
 key_break:
 pc.printf("You wrote: %s", buffer);
};
   }
```

Sources.

1. http://www.mikroe.com/downloads/get/1448/tft_proto_manual_v200.pdf
2. http://mbed.org/users/dreschpe/code/SPI_TFT_ILI9341/
3. http://mbed.org/users/dreschpe/code/SPI_TFT_ILI9341/docs/55aed13f2630/classSPI__TFT__ILI9341.html
4. http://mbed.org/users/dreschpe/code/Touch_tft/
5. http://mbed.org/users/dreschpe/code/Touch_tft/docs/ef7972c29c0e/classtouch__tft.html
6. http://mbed.org/users/dreschpe/code/TFT_fonts/
7. http://www.robotix.in/tutorials/category/avr/usart
8. https://sites.google.com/site/xbeetutorial/xbee-introduction/zigbee_setup
9. http://mbed.org/users/vcazan/notebook/mbed--xbee/
10. http://mbed.org/users/mazgch/code/SeeedStudioTFTv2/
11. http://www.libstock.com/projects/view/739/rfid-click-example
12. http://www.hoperf.com/upload/rf/RFM23BP.pdf
13. http://obex.parallax.com/object/733
14. https://github.com/StephenCarlson/KatanaLRS-code
15. http://developer.mbed.org/users/fraserphillips/notebook/mbed-gpio-pin-table/
16. https://www.django-rest-framework.org/tutorial/quickstart/
17. https://www.django-rest-framework.org/api-guide/authentication/
18. https://simpleisbetterthancomplex.com/tutorial/2018/11/22/how-to-implement-token-authentication-using-django-rest-framework.html
19. Thomas Floyed Electronics Devices
20. https://www.dimensionengineering.com/info/switching-regulators
21. https://www.microchip.com/stellent/groups/SiteComm_sg/documents/Training_Tutorials/en528032.pdf
22. https://www.avnet.com/wps/portal/abacus/resources/engineers-insight/article/understanding-esr-in-electrolytic-capacitors/
23. https://medium.com/quick-code/token-based-authentication-for-django-rest-framework-44586a9a56fb