ECE 2036 Lab 2 – Setup and Test mbed I/O Hardware

Check-Off Deadline: Section C -Tuesday Sept 23
Section B -Thursday Sept 25

Name:_______________________________

Item	TA Signoff
Part 1. (40%) Color LCD Hello World	
Part 2. (10%) Timer display on Color LCD	
Part 3. (25%) Temperature Sensor display on Color LCD	
Part 4. (25%) Three Pushbuttons controlling Speaker Tones	

Embedded devices account for 98% of the world's microprocessors. For every desktop computer, there are over 100 embedded devices. A high-end car can contain over 100 microprocessors. Embedded devices contain a computer with software that is typically not changed by the user (called Firmware). Most users are not aware that their cell phones, cameras, audio players, appliances, and TVs contain a computer with firmware. C/C++ is currently the most widely used language for embedded devices. ARM 32-bit RISC processors, which are similar to the one found in the *mbed* module, are used in about 80% of embedded devices including 95% of cell phones. There are over 30 billion ARM processors in the world!

This lab will hookup and demo new I/O hardware for your mbed ARM processor module in preparation for lab 3. This assignment will start with getting your LCD display working with the *mbed* board. In addition to your mbed module, you will need your breadboard, wire kit, and LCD module.

Color Graphics LCD Module

CAUTION

Power on pin 1 (+5V) on the Color LCD in your parts kit **uses only the 5V mbed pin VU** (i.e., **not 3.3V – mbed pin Vout**) and pin 7 (GND) is connected to the mbed GND pin. Always double check power pin connections before turning on power for the first time – if you get them wrong it might burn out the device! See the Color LCD wiki page for additional help using the Color LCD with mbed. If you want to plug the LCD directly in a breadboard, it is necessary to carefully bend over one pin as shown on the wiki page. The LCD requires several wires and if it is placed near the mbed pins used, the breadboard setup will be easier. Note that the mbed pins used (27,28,29) are on the right side of the mbed module and these pins are a bit different than the wiki page example (LCD code can use any of mbed's three Serial outputs just by changing the class constructor pin number arguments), but use them to be compatible with the embedded systems you will build in labs that follow to have space for all of the parts added later.

Instructions for Part 1: (40%)

1. Watch this Youtube video FIRST. This tells you how **NOT to break the pins** on your mBED board! **VERY IMPORTANT** - COULD WIND UP COSTING YOU $50!

Video link: http://mbed.org/blog/entry/104/

2. On your protoboard, please put the LCD on the side of the MBED chip where you have the pins p27-29. See the wiring table table below for the pin connections needed.

3. Create your *mbed* user account using the instructions in your kit. After placing the *mbed* board on the protoboard, you can connect it to your PC's USB port. It should show up on your computer as an external drive (just like a flash drive). You will put files in this drive that will be automatically be downloaded to the flash drive on your *mbed* board.

4. Import the skeleton code into your *mbed* account that is found at

http://mbed.org/users/4180_1/code/mythermostat/

5. Create a new program in your compiler environment and type in the following code. Remember … you will need to copy uLCD_4DGL.h from your skeleton code project into this new project. Cut and paste can be used in the compiler's left column that shows project source files. You can also use the import library link found on the LCD wiki page.

```
// Hello World! for the TextLCD

#include "mbed.h"
#include "uLCD_4DGL.h"

uLCD_4DGL uLCD(p28, p27, p29); // create a global lcd object

int main() {
  uLCD.printf("\nHello World!\nyourname");
}
```

6. Replace *"yourname"* above with your name. Compile this program. On some machines, this will automatically download into your *mbed* if it is connected to your computer; however, on other machines the file might be in your download folder. In this case, all you need to do is drag it to the *mbed* drive and it will be automatically downloaded to the board.

7. Pressing the reset button on the chip will automatically run the program with the most recent timestamp that is downloaded to the board. Do this and you should see your first message on the LCD display! (assuming every single jumper wire is correctly hooked up – swapping RX and TX is a common error that will prevent writing anything to the LCD)

LCD Wiring for LCD test program

Mbed	uLCD pin
5V=VU	5V (1)
Gnd	Gnd (7)
TX=P28	RX (5)
RX=P27	TX (3)
P29	Reset (9)
Note: Bend over LCD pin 10, or use LCD's cable set per wiki instructions	

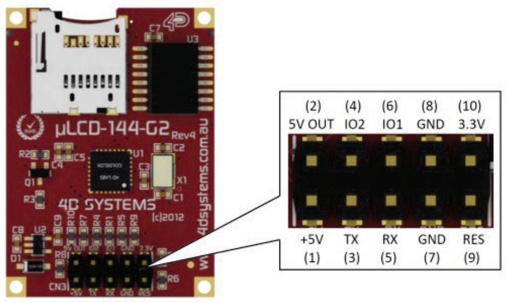


Figure 1: These are the LCD pin connections needed for this assignment. Please note that there are some pins moved between this figure and the mbed cookbook wiki demo code. Note that if you choose to use the cable, the RX and TX pins are already swapped on the cable connector label (per wiki).

Instructions for Part 2: (10%)

Please use the wiki page found at **https://mbed.org/users/4180_1/notebook/ulcd-144-g2-128-by-128-color-lcd/ and make a timer to continuously display the elapsed time in** seconds underneath your "Hello World!" message. The uLCD.locate(x,y) member function is a handy way to move the text cursor on the LCD before rewriting the time. The timer API is the easy way to start and read the timer.

Experiment with Temperature Sensor and Push Buttons

Part 3: Digital Thermometer (25%)

An TMP36 wiki page is provided that shows how to connect the TMP36 sensor and read the temperature using C/C++ on the mbed module. BE CAREFUL because it looks just like the 2N3904 transistor in the kit, so check the tiny gray marks on the case's flat area for the part number. On the wiki page sample code, note the use of C++ classes and operator overloading to read the temperature. This assignment will verify your hardware connections and sensor operation before trying your first mbed lab in a couple of weeks. Make sure the sensor is **not upside down** before connecting power the first time or you may burn it out!

Some mbeds will occasionally see a bit of noise on the temperature sensor readings. According to the data sheet, this can happen on analog data measurements when the ARM processor is in debug mode as in the mbed module setup. A bit of noise on the PC's USB 5V power might also be part of the problem. Averaging the reading several times in software and/or adding a small decoupling capacitor to filter the power supply noise and A/D input signal can help alleviate this issue, if it causes problems later. The TAs have some extra capacitors available in the lab.

Instructions For Part 3

1. In your last assignment you got your LCD working, please leave this hooked up! For a visual arrangement for your components, I would recommend the arrangement at the following YouTube site:

http://www.youtube.com/watch?v=oozKIHzeoQU

Note this video has the older BW text LCD, but you want the new color LCD in about the same location to simplify breadboard jumper wiring.
2. Now hook up the temperature sensor using the information on the wiki page at:

http://mbed.org/users/4180_1/notebook/lm61-analog-temperature-sensor/

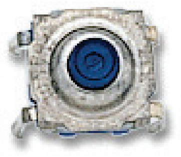
3. Create a new program and type in the following code.

```
#include "mbed.h"
#include "uLCD_4DGL.h"
//Print temperature from TMP36 analog temperature sensor
//Setup a new class for TMP36 sensor
class TMP36
{
public:
    TMP36(PinName pin);
    TMP36();
    operator float ();
    float read();
private:
//class sets up the AnalogIn pin
    AnalogIn _pin;
};

TMP36::TMP36(PinName pin) : _pin(pin)
{
// _pin(pin) means pass pin to the AnalogIn constructor
}

float TMP36::read()
{
//convert sensor reading to temperature in degrees C
    return ((_pin.read()*3.3)-0.500)*100.0;
}

//overload of float conversion (avoids needing to type .read() in equations)
TMP36::operator float ()
{
//convert sensor reading to temperature in degrees C
    return ((_pin.read()*3.3)-0.500)*100.0;
}

//use the new class to set p15 to analog input to read and convert TMP36 sensor's
voltage output
TMP36 myTMP36(p15);

//also setting unused analog input pins to digital outputs reduces A/D noise
//see http://mbed.org/users/chris/notebook/Getting-best-ADC-performance/
DigitalOut P16(p16);
DigitalOut P17(p17);
DigitalOut P18(p18);
DigitalOut P19(p19);
DigitalOut P20(p20);

uLCD_4DGL uLCD(p28, p27, p29); // create a global lcd object

int main()
{
    float tempC, tempF;

    while(1) {
        tempC = myTMP36.read();
        //convert to degrees F
        tempF = (9.0*tempC)/5.0 + 32.0;
        //print current temp
        uLCD.locate(2,2);
        uLCD.printf("%5.2F C %5.2F F \n\r", tempC, tempF);
        wait(.5);
    }
}
```

Part 4: Three Button Keyboard (25%)

In the second part of this assignment you will build a basic 3-key musical keyboard. Do not take off the other components! The point of this exercise is to build up and verify your components to help with your future mbed lab. I would advise that you try to get the keys working first with some of the LEDs and then add functionality with the speaker.

Using Pushbuttons



Read the pushbutton wiki page and watch the videos for additional help using pushbuttons with mbed. Small pushbuttons in the kit are available for use on your breadboard. Look at the constructors in the sample code to determine the pin connections to the mbed board. Note that the pushbutton needs to be plugged in just like an IC (i.e., flat part of pins on sides) to avoid shorting the switch.

Using the Speaker

Use the driver transistor and speaker to make different tones when each button is pressed. See the speaker wiki page for additional hardware and software help using speakers with mbed. Look at the constructors in the sample code to determine the pin connections to the mbed board. The jumper wires will need to be connected underneath the speaker before it is plugged into the breadboard.

Instructions For Part 4

1. Please leave all previous component hooked up on your board (i.e. LCD and temperature sensor)

2. Now hook up three push buttons using the information on the wiki page at:

http://mbed.org/users/4180_1/notebook/pushbuttons/

I would suggest testing the code on the following page without the PlayNote functions to see if the push buttons work with the LEDs built into the mbed board.

3. Now hook up the speaker and transistor driver using the information on the wiki page found at:

https://mbed.org/users/4180_1/notebook/using-a-speaker-for-audio-output/

4. Create a new program and type in the following code (Don't forget to add the *Speaker* and *PinDetect* include (*.h) source code files to the new project) :

```
#include "mbed.h"
#include "Speaker.h"
#include "PinDetect.h"

DigitalOut myled1(LED1);
DigitalOut myled2(LED2);
DigitalOut myled3(LED3);
DigitalOut myled4(LED4);

PinDetect pb1(p23);
PinDetect pb2(p24);
PinDetect pb3(p25);

// setup instance of new Speaker class, mySpeaker using pin 21
// the pin must be a PWM output pin
Speaker mySpeaker(p21);
// Callback routine is interrupt activated by a debounced pb1 hit
void pb1_hit_callback (void)
{   // CODE HERE WILL RUN WHEN INTERUPT IS GENERATED
myled1 = !myled1;
mySpeaker.PlayNote(200.0,0.25,0.1);
}
// Callback routine is interrupt activated by a debounced pb2 hit
void pb2_hit_callback (void)
{  // CODE HERE WILL RUN WHEN INTERUPT IS GENERATED
myled2 = !myled2;
mySpeaker.PlayNote(400.0,0.25,0.1);
}
// Callback routine is interrupt activated by a debounced pb3 hit
void pb3_hit_callback (void)
{  // CODE HERE WILL RUN WHEN INTERUPT IS GENERATED
myled3 = !myled3;
mySpeaker.PlayNote(800.0,0.25,0.1);
}
int main()
{
//setup three SPST push buttons
    pb1.mode(PullUp); //add internal pullup resistor
    pb2.mode(PullUp);
    pb3.mode(PullUp);
    // Delay for initial pullup to take effect
    wait(.01);
    // Setup Interrupt callback functions for a pb hit
    pb1.attach_deasserted(&pb1_hit_callback);
    pb2.attach_deasserted(&pb2_hit_callback);
    pb3.attach_deasserted(&pb3_hit_callback);
    // Start sampling pb inputs using interrupts
    pb1.setSampleFrequency();
    pb2.setSampleFrequency();
    pb3.setSampleFrequency();
// pushbuttons now setup and running
    while(1)
    {
     myled4 = !myled4; //blink to show code running
     wait(0.5);
    }
} //end main
```