**ECE 2036 Lab 3 Prelab – Setup and Test I/O Hardware**

Check-Off Deadline: Section A -Monday Oct 7th

Section B -Tuesday Oct 8th

 Section C -Wednesday Oct 9th

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| **Item** | **TA Signoff** |
| Part 1. (40%) Text LCD Hello World |  |
| Part 2. (10%) Timer display on LCD |  |
| Part 3. (25%) Temperature Sensor display on 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 up to 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, and TVs contain a computer with firmware. C/C++ is currently the most widely used language for embedded devices. ARM processors, which are similar to the one found in the *mbed* module, are used in about 80% of embedded devices including most cell phones.

This assignment will start with getting your LCD display working for the *mbed* board. In addition to your mbed module, you will need your wire kit, LCD board, and a 1K resistor.



**CAUTION**

Power on pin 2 VCC on the text LCD in your parts kit **uses 5V mbed pin VU** (i.e., **not 3.3V – mbed pin Vout** as shown on the mbed LCD wiki page) and the 1K contrast resistor is needed. See the [Text LCD wiki page](http://mbed.org/cookbook/Text-LCD) for additional help using the text LCD with mbed. More specific details on the text LCD in your kit are listed on the [mbed inventor’s kit web page](http://users.ece.gatech.edu/~hamblen/mbed/mbed_inventors_kit.htm). The LCD requires several wires and if it is placed near the mbed pins used, the breadboard setup will be easier. Figure 1 in this homework gives you the exact pinouts that correspond to the given code, these pins may be different than the wiki page, but I would require that you use them to be compatible with the embedded systems you will build in your labs.

**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!

Video link: http://www.youtube.com/watch?v=c9n7bxpqWmg

**2.** On your protoboard, please put the LCD on the side of the MBED chip where you have the pins p22-p27. For a visual, I would recommend the arrangement at the following YouTube site (<http://www.youtube.com/watch?v=oozKIHzeoQU>). See Figure 1 for the pin connections needed.

**3.** Create your *mbed* account using the instructions in your kit. After placing the *mbed* board on the protoboard, you can connect this to your USB port. It should show on your computer as an external drive. You will put files in this drive that will be automatically be downloaded to the flash drives on your *mbed* board.

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

<http://mbed.org/users/4180_1/code/mymbedthermostat/>

**5.** Create a new program in your compiler environment and type in the following code. Remember … you will need to copy the TextLCD.h from your skeleton code.

**// Hello World! for the TextLCD**

**#include "mbed.h"**

**#include "TextLCD.h"**

**TextLCD lcd(p22, p23, p24, p25, p26, p27); // create a global lcd object**

**int main() {**

 **lcd.printf("Hello World!\n");**

**}**

**6.** 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!

**Instructions for Part 2: (10%)**

Please use the wiki page found at [mbed.org/cookbook/Text-LCD](http://mbed.org/cookbook/Text-LCD) and make a timer to display elapsed time in seconds underneath your “Hello World!” message.

**Figure 1:** These are the pin connections needed for this assignment. Please note that there are some differences between this figure and the mbed cookbook online help.

**Experiment with Temperature Sensor and Push Buttons**

**Part 3: Digital Thermometer ( 25%)**



An [TMP36 wiki page](http://mbed.org/users/4180_1/notebook/lm61-analog-temperature-sensor/) 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 marks on the case’s flat area for the part number. This assignment will verify your hardware connections and sensor operation before trying your first mbed lab in a couple of weeks.

**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>

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 "TextLCD.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);

TextLCD lcd(p22, p23, p24, p25, p26, p27); // rs, e, d4-d7

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

 lcd.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.

**Pushbuttons**



Read the [pushbutton wiki page](http://mbed.org/users/4180_1/notebook/pushbuttons/) and watch the videos for additional help using pushbuttons with mbed. Small pushbuttons are available for use on your breadboard. I would like for you to look at the constructors in the sample code to determine the pin connections to the mbed board.

**Speaker**



Use the driver transistor and speaker to make tones when each button is pressed. See the [speaker wiki page](https://mbed.org/users/4180_1/notebook/using-a-speaker-for-audio-output/) for additional hardware and software help using speakers with mbed. I would like for you to look at the constructors in the sample code to determine the pin connections to the mbed board.

**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.

#include "mbed.h"

#include "Speaker.h"

#include "PinDetect.h"

DigitalOut myled1(LED1);

DigitalOut myled2(LED2);

DigitalOut myled3(LED3);

DigitalOut myled4(LED4);

PinDetect pb1(p28);

PinDetect pb2(p29);

PinDetect pb3(p30);

// 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 myled1 = !myled1;

myled3 = !myled3;

mySpeaker.PlayNote(800.0,0.25,0.1);

}

int main()

{

//setup push buttons

 pb1.mode(PullUp);

 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;

 wait(0.5);

 }

} //end main