

## ECE 2036 Lab 2 – Introduction to mbed; Thermometer and Stopwatch

Check-Off Deadline:

Section A – Wednesday Feb 10

Section B – Tuesday Feb 9

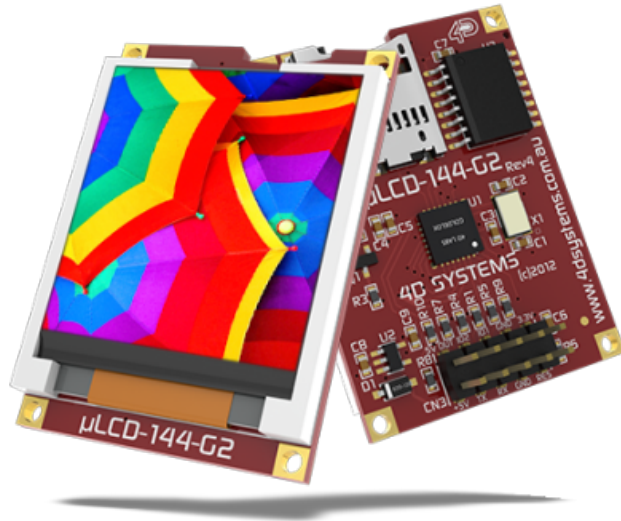
Name: \_\_\_\_\_

Print this first page and bring it to a TA or grader when your lab is complete. Also bring your assembled breadboard, components, and laptop. You will demonstrate the operation of your completed lab for the TA or grader. Show your code to the TA or grader, and expect to be asked a few questions that test your understanding of your code. Grades are determined based on the table below. Keep this sheet for your records after your grade has been assigned. The TA or grader will record your score on their own spreadsheet.

Item	TA Signoff
Part 1. (50%) Color LCD Hello World	
Part 2. (25%) Temperature Sensor display on Color LCD	
Part 3. (25%) Stopwatch performs correctly with LCD and pushbutton	
All checkoffs complete by 5 pm on Feb 5 (+5%)	
Late checkoffs (-20% per day late, weekends count as one day)	
Artistic Merit (up to TA, +5% max)	

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 200 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! There are two ARM processors on your *mbed* module, one is just used for the USB interface to the main processor. The color LCD has another processor.

In this lab, you will hook up and demo new I/O hardware for your *mbed* ARM processor, and construct a simple thermometer and stopwatch. 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 color 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! Be sure to read the [Color LCD wiki page](#) for additional help (including photos) for using the Color LCD with mbed. If you want to plug the LCD directly into the 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 these pins to be compatible with the embedded systems you will build in labs that follow. We need to save space for all of the parts added later.

### **Part 1: Hello World (50%)**

Instructions:

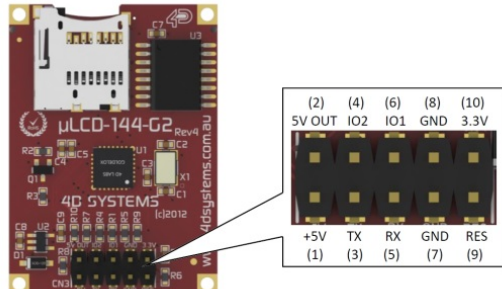
**1.** Watch this Youtube video FIRST. This tells you how **NOT to break the pins** on your mBED board! **VERY IMPORTANT** - COULD WIND UP COSTING YOU \$50!

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 that follows for the pin connections needed.

**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)	
<b>Note:</b> Bend over LCD pin 10, or use LCD's cable set per the 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 LCD wiki page wiring table and photos).

3. Create your *mbed* user account using the instructions in your mbed box. After placing the *mbed* board on the protoboard, you can connect it to your PC's USB port, which is needed to setup a new account. The plugged-in mbed board should show up on your computer as an external drive (just like a USB flash drive). When you write a program and compile it, there will be a binary executable file created. This file will have the extension *.bin*. You will then move these *\*.bin* files to the mbed external drive, using drag-and-drop or another method of your choice. Doing this will be download your programs to the flash drive to run on your *mbed* board.

4. To quickly get all of the include files needed later, after logging onto mbed.org, "Import" the skeleton code into your *mbed* account's project space that is found at

[http://mbed.org/users/4180\\_1/code/mythermostat/](http://mbed.org/users/4180_1/code/mythermostat/)

5. Next, create a new program in the mbed web compiler environment. You will need to copy uLCD\_4DGL.h from your skeleton code project into this new project in order to use the LCD. Cut and paste can be used in the compiler's left column, which shows project source files. You can also use the import library link found on the LCD wiki page. Then type the following code into the main.cpp file of your new program.

```
// Hello World! for the TextLCD
#include "mbed.h"
#include "uLCD_4DGL.h"

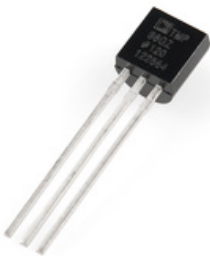
uLCD_4DGL uLCD(p28, p27, p29); // create a global lcd object

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

6. Replace “*yourname*” above with your name. Compile this program with no errors and a \*.bin file is created to run the code. On some machines, this will automatically download into your *mbed* drive if it is connected to your computer; however, on other machines or web browsers 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 mbed will automatically run the program (\*.bin file) 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). If they are swapped, after the LCD times out with no data, a splash screen appears.

## Part 2: Digital Thermometer ( 25%)



An [TMP36 wiki page](#) is provided that shows how to connect the analog TMP36 sensor (to p15) 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 next mbed lab in a couple of weeks. Make sure the sensor is **not upside down** (per wiki) 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:

1. In part 1 you got your LCD working, please leave this hooked up! For a visual arrangement for your components, use something like the arrangement at the following YouTube site:

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

Some other parts are added in later labs on the right side of the mbed, so leave space for them.

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

3. Create a new program and type in or cut and paste the following code in main.cpp that will display the temperature reading from the sensor on the LCD.

```
#include "mbed.h"
#include "uLCD_4DGL.h"
#include "TMP36.h"

//Instantiate your TMP36 analog temperature sensor, connected to pin 15
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 3: Stopwatch (25%)**

In the last part of this assignment you will make a stopwatch on the LCD that starts, stops, and resets using a pushbutton. Do not take off the other components! The point of this exercise is to build up and verify your components work to help with your future mbed labs.

#### **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., note flat part of pins on sides) to avoid shorting the switch.

Instructions:

1. Please leave all previous components hooked up on your board (i.e. LCD and temperature sensor) – It will save you time later!
2. Now hook up a push button to pin 25 using the information on the wiki page at:

[http://mbed.org/users/4180\\_1/notebook/pushbuttons/](http://mbed.org/users/4180_1/notebook/pushbuttons/)

3. Start a new program to experiment with the pushbutton. Be sure to copy the PinDetect library into the program folder. Type in and run the following test code into main.cpp.

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

// Instantiate LEDs myled1-4 (objects of class DigitalOut)
DigitalOut myled1(LED1);
DigitalOut myled2(LED2);
DigitalOut myled3(LED3);
DigitalOut myled4(LED4);

// Instantiate pushbutton pb1 (object of class PinDetect) on pin 25
PinDetect pb1(p25);

// global variable LED_run is true if the LEDs should be flashing
bool volatile LED_run=true;
```

```

// This is the function that will be called when an interrupt
// is generated by pushing the pushbutton
void pb1_hit_callback (void)
{
// switch the value of LED_run
LED_run = !LED_run;
}

// main function
int main() {
// These are member functions of class PinDetect
// We are calling them in order to set up pushbutton pb1
// First we add an internal pullup resistor
pb1.mode(PullUp);
// small delay to allow previous command to take effect
wait(0.01);
// Specify the interrupt callback function
// This command specifies that the function pb1_hit_callback
// will be called when the pb1 is pushed
pb1.attach_deasserted(&pb1_hit_callback);
// Start sampling the pb1 input using interrupts
pb1.setSampleFrequency();

// infinite loop
while(1) {
// see if you can figure out what this will do
// and run the code to see if you were right!
    if (LED_run) {
        myled4 = 0;
        myled1 = 1;
        wait(0.2);
        myled1 = 0;
        myled2 = 1;
        wait(0.2);
        myled3 = 1;
        myled2 = 0;
        wait(0.2);
        myled4 = 1;
        myled3 = 0;
        wait(0.2);
    }
    else {
        myled1 = 0;
        myled2 = 0;
        myled3 = 0;
        myled4 = 0;
        wait(0.5);
    }
}
}

```

4. Using everything you've learned, write a new program to implement a combined thermometer and stopwatch. Your LCD screen should continuously display the temperature, as implemented in the previous step. In addition, beneath the temperature data should be a field saying "Stopwatch time: 0". If the user presses the pushbutton when the number in that field is zero, that number should start to increment by 1 every second. If the user presses the pushbutton while that number is increasing, that number should stop increasing. If the user pushes the pushbutton when that number is not increasing but is greater than zero, that number should reset to 0.

For extra credit you may use artistic license to make the display more attractive, but your display must show the temperature in Fahrenheit and Celsius and the stopwatch time in an immediately understandable and user-friendly way.