

CG2271 Real Time Operating system AY 2017

Lab 2 – I/O Interfacing (6% of your final grade)

Lab Lessons: Wednesday 13 and 20 Sept

IVLE Submission of answer book: Friday 22 Sept 2359

Demo: Wednesday 4 Oct

1. Introduction

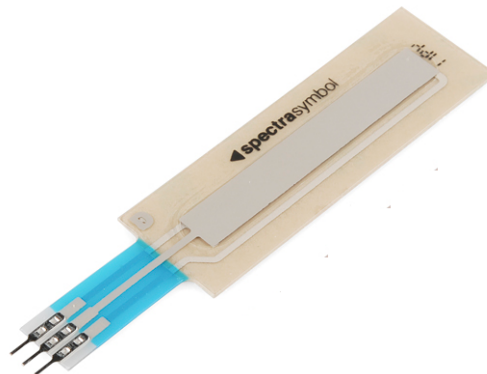
In the previous lab you learnt how to use Arduino Software with Eclipse IDE. In this lab we will build a simple circuit to explore ways that we can perform I/O interfacing with sensors and actuators.

This lab is worth 45 marks for the answer book and 15 marks for the demo for a total of 60 marks.

2. Circuit for This Lab

- i. Wire up a pushbutton at digital pin 2 on the Arduino. This pushbutton should pull pin 2 low when pressed. Pin 2 should be pulled high when the button is not pressed. This is called a pull-up resistor. For our circuit we use the 10K resistor for pull-up. Note: You should never leave a digital input “floating”, i.e., it must be pulled up when the pushbutton is not pressed. (See <https://learn.sparkfun.com/tutorials/pull-up-resistors> for more details on pull-up resistors)
- ii. Similarly connect another pushbutton to digital pin 3 along with a pull-up resistor.
- iii. Connect a potentiometer to analog input A0.
- iv. Connect one LED to digital pin 6 and one LED to digital pin 7. **Use the 330 ohm resistors to limit current through the LEDs.** Failing to use a current limiting resistor will cause the device to be irrecoverably damaged.

In addition your Aduino kit comes with a touch sensor that looks like this:



- v. Connect this to Analog Input A1 in the same way that you connect a potentiometer. If you don't know how to use a potentiometer, read this:
<http://www.arduino.cc/en/Tutorial/Potentiometer>.

Question 1 (6 marks)

Sketch your circuit in the Answer Book. You can use a tool like Fritzing (<http://fritzing.org/download/>) to help you. Attach only the schematic view of the circuit in the Answer Book. Make sure the diagram is clean and easy to read.

3. Adjusting the Range of the Potentiometer

We now want to program our Arduino so that we take a reading from the potentiometer and use the value to control the brightness of the LED.

Create a new project and key in the following program:

```
/****** SAMPLE PROGRAM 2 *****/  
  
#include <Arduino.h>  
  
#define PIN_LED 6  
#define PIN_PTTM 0  
  
void setup() {  
    pinMode(PIN_LED, OUTPUT);  
}  
  
void loop() {  
    int val;  
    val = analogRead(PIN_PTTM);  
    analogWrite(PIN_LED, val);  
    delay(500);  
}
```

Compile and then download your program. To test your program, turn the potentiometer. You will find that the potentiometer controls the brightness of the LED connected to pin 6.

Question 2 (3 marks)

Turn the potentiometer all the way to the left. Then turn it all the way to the right. Notice that rather than going from fully dark to fully bright as you turn the potentiometer, the LED actually goes from dark to bright then back to dark, and then back to bright, etc.

This behaviour is caused by the ADC connected to the potentiometer returning a value between 0 and 1023, while the PWM generator at pin 6 can only understand values of between 0 and 255. Explain why this mismatch causes the behaviour observed in this question.

Question 3 (5 marks)

(NOTE: You are *not allowed* to use any existing Arduino library functions to solve this question. You may however use any standard C library call, like those you learnt in CS1010)

Write a function called “remap” with the following parameters to remap the values return by the ADC from 0 to 1023, to 0 to 255. Modify your code to use this function. Turn the potentiometer all the way to the left. Then turn it all the way to the right. Does the LED brightness change appropriately?

Cut and paste your “remap” function to your answer book and explain why it solves the problem in Question 2. Please note that the code shown for remap in the lecture may not be fully functional.

```
int remap(int val);
```

4. Adjusting the Range of the Touch Sensor

Adjusting the touch sensor is trickier than the potentiometer. This is because the potentiometer will generally return a value of between 0 and 1023 (your own potentiometer might vary slightly but not significantly).

The touch sensor on the other hand returns dirty data. To do the remapping properly you need to know the minimum and maximum values returned by the touch sensor. To figure this out, we first write a program that dumps out the potentiometer and touch sensor values on a serial port of the PC.

Create a new project and key in the following program:

```
/****** SAMPLE PROGRAM 2 *****/
#include <Arduino.h>

#define PIN_PTTM 0
#define PIN_TOUCH 1

void setup() {
    // initialize serial communications at 115200 bps:
    Serial.begin(115200);
}

void loop() {
    int val, touch;

    // read potentiometer's value
    val = analogRead(PIN_PTTM);
    // read touch sensor's value
    touch = analogRead(PIN_TOUCH);

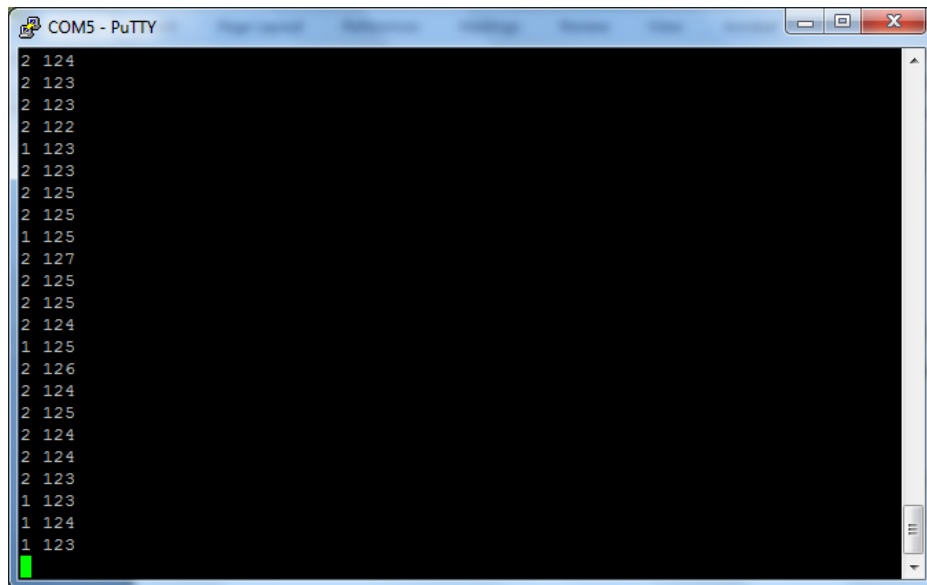
    // dump them to serial port
    Serial.print(val);
    Serial.print(" ");
    Serial.print(touch);
    Serial.println();

    // 200ms pause
    delay(200);
}
```

Compile and then download your program. Now

- i. Start up a terminal program like puTTY or the “Serial Monitor” in the Arduino environment.
- ii. Set up your terminal program so that it reads from the Arduino’s COM port at a rate of 115200 bps.

You will see that your program on the Arduino is returning two columns of numbers as shown next:



Turn the potentiometer. You will notice the numbers in the first column changing. Similarly touch the touch sensor, and you will see the numbers in the second column changing. What is happening here is that your program is using the Serial class to write values read to your computer through the COM port. The Serial class is an extremely useful tool for debugging programs on the Arduino.

Question 4 (3 marks)

Place your finger at various points on the touch sensor, both at the extreme top of the sensor and extreme bottom, and points in between. Write down the maximum and minimum values you observed.

Question 5 (5 marks)

Write a function called “remapTouch” that remaps the maximum and minimum values you observed for the touch sensor to the range of 75 to 450. The prototype for remapTouch is:

```
int remapTouch(int value);
```

Cut and paste your “remap” function to your answer book.

Question 6 (6 marks)

Create a new project and copy and paste the program you have written for question 3 (changing LED brightness with potentiometer). Modify your program in Question 3 so that the touch sensor now controls how quickly the LED connected to pin 7 **blinks** (repeated on and off) using the digital I/O Pin 7. The minimum value passed to this function should be 75 and the maximum value should be 450, i.e. the smallest delay value should be 75ms and the largest 450ms.

Cut and paste your code into your answer book.

The final result should be that the potentiometer controls the brightness of the LED on pin 6 and touch sensor controls the speed of blinking of the LED connected on pin 7.

5. Using Interrupts in the Arduino

The code below shows a very simple example of how to use interrupts. This example uses INT0 (digital pin 2) to switch an LED on and off. An ISR (called isr()) is created to respond to INT0. The interrupt is configured to respond to a LOW->HIGH transition at digital pin 2.

```
/****** SAMPLE PROGRAM 3 *****/  
  
#include <Arduino.h>  
  
unsigned char flag=0;  
  
void isr()  
{  
    flag=!flag;  
}  
  
void setup()  
{  
    attachInterrupt(0, isr, RISING);  
    pinMode(6, OUTPUT);  
}  
  
void loop()  
{  
    digitalWrite(6, flag);  
}
```

One thing about this program: It works fine *in theory*, but if you actually upload this program to the Arduino, you will find that the LED does not light up reliably. This is because the switch is mechanically imperfect and causes a phenomenon called “bouncing”.

Question 7 (6 marks)

Modify Sample Program 3 to ensure de-bouncing of the switch. This is done by adding delay so that the switch is not read during the bouncing stage.

Cut and paste your modified code into your answer book.

Question 8 (11 marks)

In question 6, the LEDs respond immediately to changes in the potentiometer or touch sensor. So turning the potentiometer will cause the LED at pin 6 to brighten or darken, and moving your finger along the touch sensor will change the rate at which the LED at pin 7 is blinking.

We want to modify our program so that the brightness of the LED at pin 6 stop changing with potentiometer after the button at pin 2 is depressed once (and anytime again from there on regardless of the state of the button), and the rate of blinking of the LED at pin 7 stop changing with touch sensor after the button at pin 3 is depressed once (and anytime again from there on regardless of the state of the button). You need to write two interrupt service routines corresponding to the two push buttons; the ISRs should note down the button depression events through global flags that should be used in the code within the main loop. Ensure that the input from both the switches are de-bounced.

6. Demo Session

Submit the Lab 2 answer book to IVLE Files → Lab2-Submissions. Please ensure that you submit your file as GXX.doc or GXX.pdf (XX is your group number). You should make one submission per team.

You will demonstrate Question 7 and Question 8 to your lab TA at the start of your lab session for Lab 3. Please be punctual for your demonstration. Your demonstration and oral Q&A is worth 10 and 5 marks respectively.