Name: *Roll Number:*

Laboratory 1 - Introduction to Microprocessors: 'Hello World'

Introduction:

This assignment introduces you to the basic tools and devices that you will use throughout the rest of the semester in this lab:

1. Arduino microcontroller board assembly. The board contains the ATMega328 microcontroller and several other components that make it a self-contained package which can be interfaced directly to other hardware and programmed from a PC.
2. You will also learn how to interface the board to the PC over a USB connection and work on writing a simple program for the ATMega328 microcontroller. The programs are written in a language very much like C using an Integrated Development Environment (IDE). The IDE setup on your laptop has been covered in the hands-on tutorial session.

Procedure:

This lab consists of the following parts:
Part A: Get familiar with the hardware you will use. Answer several questions based on your study of the Arduino board and IC datasheet and setup + test the IDE on your PC/laptop.
Board setup: Setup your Arduino hardware board by connecting it to your laptop.
Part B: Digital I/O on the Arduino. Write the program specified in the Arduino IDE on your laptop. Connect your board to the PC and setup the IDE. Compile the program and download to the board.
Part C: Analog I/O on the Arduino. Test the 6 analog pins on the Arduino by writing the program specified.
Instructions for format of submitting the assignment solutions on Moodle are given page 3

Part A: Hardware [2]

A.1) What is the clock frequency for the ATMega328 chip as setup on your Arduino board? Examine your Arduino board carefully – component names and values are printed on the board, and they can be cross-referenced with the Atmega328 datasheet.
A.2) What is the maximum safe clock frequency you can use?
A.3) What limits the maximum operating frequency? Give appropriate graphs from the datasheet.

A.4) Board Setup: (covered in Hands-on tutorial session, to be done independently for your hardware) 2

The Arduino board is programmed over USB and has two power-supply options: (a) power is drawn from a 9V power supply or (b) power is drawn from the USB cable. Unless specified otherwise, we usually use the USB cable to supply power as well as
Connect the board to the PC over USB.
Start up the Arduino Integrated Development Environment (IDE) on the PC and familiarize yourself with it. When you connect your Arduino board to the PC USB port, you should see the RX/TX LED's next to the USB connector on the board flash, indicating that a connection has been established. Two settings are necessary for the IDE to connect to to the board:

1. In Tools→Serial Port, you should select the menu item 'COMn' (Windows) or /dev/ttyACMn (Linux) when the Operating System's USB driver will detects the board connection. n may change every time the board is reconnected.
2. The board ID in Tools→Board menu must be set to 'Arduino UNO'

Provide a screenshot of your board successfully connected to your PC/Laptop.

Name: *Roll Number:*

Part B: Digital I/O [10]

Objective: Determine the delays between successive C directives in your program.

As discussed in the tutorial session, a high level program instruction like:

```
digitalWrite(13, HIGH);    // Set pin 13 output to logic level HIGH
```

translates into a sequence of binary opcodes after compilation. Execution of these opcodes occurs over multiple clock cycles.

In certain time critical applications (for example, controlling a stepper motor, or steering a quadrotor), it is necessary to account for such short delays even if they are at the microsecond level.

QUESTION B.1: **2**

If you had access to a DSO, how would you determine the number of CPU clock cycles required for the above instruction to be executed?

Write a short program segment implementing your method, and indicate with a comment what hardware connections you would need to make such a measurement.

QUESTION B.2: **8**

Without access to a DSO, you will often need to execute `Serial.print()` statements to send current values of variables back to the host PC so you can debug your program.

As discussed in the intro session, any communications over the USB – Serial connection add an extra time delay to the program execution.

Objective: Determine the time delay introduced in program execution by making a function call `Serial.println(…)`

Notes:

There are several time scales in this setup:

(1) The Atmega328 clock running on the Arduino PCB (whose frequency you have determined in Part A
(2) The much higher frequency clock (~ GHz) running on the motherboard of your laptop
(3) The 'wall time' – this is the physical time that your laptop shows (just like your wristwatch or smartphone) usually synchronized over the internet.

We are **NOT** interested in the absolute time as measured by (1), (2) or (3) above.

QUESTION: Determine the time delay ΔT when you execute the following sequence of function calls:

```
1) digitalWrite(pinX, LOW);  // t=0, pinX=13 is in setup()
2) Serial.print(0);
3) digitalWrite(pinX, HIGH); // Drive pinX output to high (5V)
4) Serial.print(1);
5) Serial.print("Game over"); // t = ΔT // confirm to user
```

Note: there is no relativity involved here! Think ahead to a situation where you are using the digital signal emitted by pinX to trigger some hardware switch. How much of a ΔT delay must you account for, between the start of the program fragment above and the friendly 'Game over' instruction being conveyed to the person looking at the Arduino Serial Monitor window?

If you had a DSO it would be straightforward to determine real ΔT as per part (B.1). However, in the coming assignments, you will often need to debug your code using such `Serial.print()` function calls. The objective here is to benchmark what is the extra timing delay introduced by doing so.

You will need:

- ➢ A method to directly peek at the data coming in over the Serial port – dig into the low level software details of how Serial communication works between the Arduino and the host PC.
- ➢ At the PC end, the Serial monitor window in Arduino IDE has a provision for putting *ms* accuracy timestamps on the incoming data. But is this good enough? The Serial monitor window is a high level Java software program involving lots of timing overhead!
- ➢ Experimentally demonstrate your results with appropriate screenshots.

Name: *Roll Number:*

Part C: Analog Input [8]

The Arduino has 6 analog input pins on the PCB labelled 0..5.
The objective is to make sure that analog pins `A0, A1 … A5` in your program code when you make the function call `analogRead(A..)` correspond to the physical pins 0,1,2,3,4,5 respectively marked on your PCB. Write a program that reads the voltage levels on each of these pins. The measured values are sent to the host PC over USB-serial using `Serial.print(..)` function calls for verification

C.1) 4

In absence of an independent voltage source, use the parts available in your electronics kit to get an assortment of analog voltage levels and test your program.
Draw a circuit diagram, and include a picture of your setup in the solution

C.2) 4

Write your program code. Provide a picture of your hardware setup including the breadboard and screenshot of the running program with appropriate measurements.

Practical notes on submitting solutions:

1. Solutions to each assignment are like a 'mini project report' to be submitted on Moodle.
2. Solutions must be in PDF as a single file, with your name roll number and page number at the top of each page
3. Solutions to each part must be labelled clearly as A.1), A.2) etc – start each sub-question on a new page
4. Neat circuit diagrams are to be drawn with the free online utility http://www.circuitlab.com (or some equivalent software like LTSpice)
5. Setup a simple method to take screenshots of your working code (you will be making lots of them in this semester!) One method is to use the '`PrtScr`' button on your keyboard. Depending on the model of your computer keyboard it may be invoked by a key combination `'Fn+PrtScr'` This copies your screenshot to the clipboard. Then you can paste it into whatever word processing program you are using. *Whatever your chosen method, the screenshot must be identifiably unique to your solution – include your username (in Linux), or the currently logged in User in Windows/MacOS*