EE3422- Embedded and Realtime Systems
LAB-1

Section 1: Introduction to Arduino Programming

The Arduino (Uno) Board is a micro-controller board that was created to house the ATmega328 chip. The chip is a high performance and low power 8-bit micro-controller that has 23 programmable I/O lines, 32K bytes of flash memory (of which 0.5KB is already used for the Boot loader), 1k bytes of EEPROM and 2k bytes of RAM. The Arduino Uno board provides the user with 6 analog input pins, 14 digital I/O pins of which 6 of them can also be used for PWM outputs, a power jack, a USB port, an ICSP header, a reset button, a small LED connected to digital pin 13, and a 16MHz crystal oscillator. (See Arduino Uno Component View for more details about the parts.)

1.1 Arduino UNO Component View

A pictorial view of the Arduino Uno board's peripherals can be found in the Figure 1.

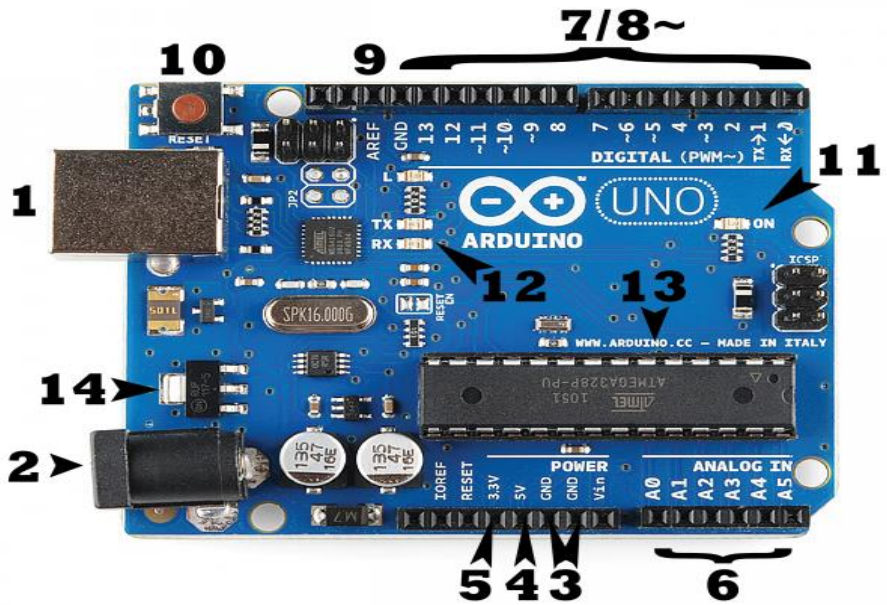

Figure 1: Arduino Uno board pictorial view

Description of highlighted parts of Arduino Uno

1- **USB Port:** allows the user to connect using a USB cable the board to a PC to upload sketches or provide a voltage supply to the board. This is also used for serial communication through the serial monitor from the Arduino software.
2- **DC Jack:** where the power source (AC-to-DC adapter or battery) should be connected. It is limited to input values between 6-20V but recommended to be around 7-12V.
3- **Ground:** to provide ground to your circuit.
4- **5V:** Power Pins.
5- **3.3V:** 3.3V pin.
6- **Analog in:** pins (A0-A5) that take-in analog values to be converted to be represented with a number in the range 0-1023 through an Analog to Digital Converter (ADC).
7- **8: Digital Input/ Output:** Input and output pins (0-13) of which 6 of them (3, 5, 6, 9, 10 and 11) also provide PWM (Pulse Width Modulated) output by using the analogWrite() function. Pins (0 (RX) and 1 (TX)) are also used to transmit and receive serial data.
9- **Aref:** For external reference voltage.
10- **Reset pin:** a button that is pressed whenever you need to restart the sketch programmed in the board.
11- **Power LED Indicator:** LED that lights up when the board is connected to a power source.
12- **Tx Rx LEDs:** Pins (0 (RX) and 1 (TX)) are used to transmit and receive serial data.
13- **Main IC:** 8-bit microcontroller that processes the sketch you programmed.
14- **Voltage Regulator IC:** Converts input DC voltage to 5v and 3.3v.

1.2 Arduino Programming Fundamentals

Arduino program is known as **Sketch**. It's the unit of code that is uploaded to and run on an Arduino board. The basic outline of each program consist of three main sections:

a. Variables

A variable is used to store a piece of data. A variable has a name, type and a value. An example of a variable having name **led1**, type **integer** and value shown below.

```
int led1 = 7;          // LED1 connected to digital pin 7
```

Variables can be global or local depending on the choice of user. Global variables can be accessed throughout the sketch and in all functions. Global variables are usually defined in the start of sketch. Whereas, local variables that has their significance only in the function where they are initialized. The advantage of using a variable is that it's easier to move the LED to a different pin: you only need to edit the one line that assigns the initial value to the variable.

b. The "Setup"

```
void setup()
```

The **setup**() function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

c. The "Loop"

```
void loop()
```

After creating a setup() function, which initializes and sets the initial values, the **loop**() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond.

1.3 Functions and Libraries

Segmenting code into functions allows a programmer to create modular pieces of code that perform a defined task and then return to the area of code from which the function was "called". The typical case for creating a function is when one needs to perform the same action multiple times in a program. An example of a function is mentioned below, where **int myMultiplyFunction(int x, int y)** is used to multiply two input variables **x** and **y** and return the **result** to main void **loop**() in the variable **k**.

```
void setup(){
  Serial.begin(9600);
}

void loop() {
  int i = 2;
  int j = 3;
  int k;

  k = myMultiplyFunction(i, j); // k now contains 6
  Serial.println(k);
  delay(500);
}

int myMultiplyFunction(int x, int y){
  int result;
  result = x * y;
  return result;
}
```

Sometimes you need to add libraries while using LCDs, servo motors and sensors etc. Arduino libraries are a convenient way to share code such as device drivers or commonly used utility functions. Additional explanation on how to add libraries to your code and how they work is discussed in Section 2.

Section 2: Tasks to Perform

2.1 An example – Digital Input and Output

Digital input and output are the most fundamental physical connections for any microcontroller. Arduino Uno has 14 digital input and output pins (0-13). These pins can be configured as input or output by initializing **pinMode(Pin, INPUT);** for input or **pinMode(Pin, OUTPUT);** for output.

You need to construct the circuit shown in Figure 2 and download the code (DigitalInputOutput.ino) from Moodle. Please go through the program and analyse the functionality of the program.

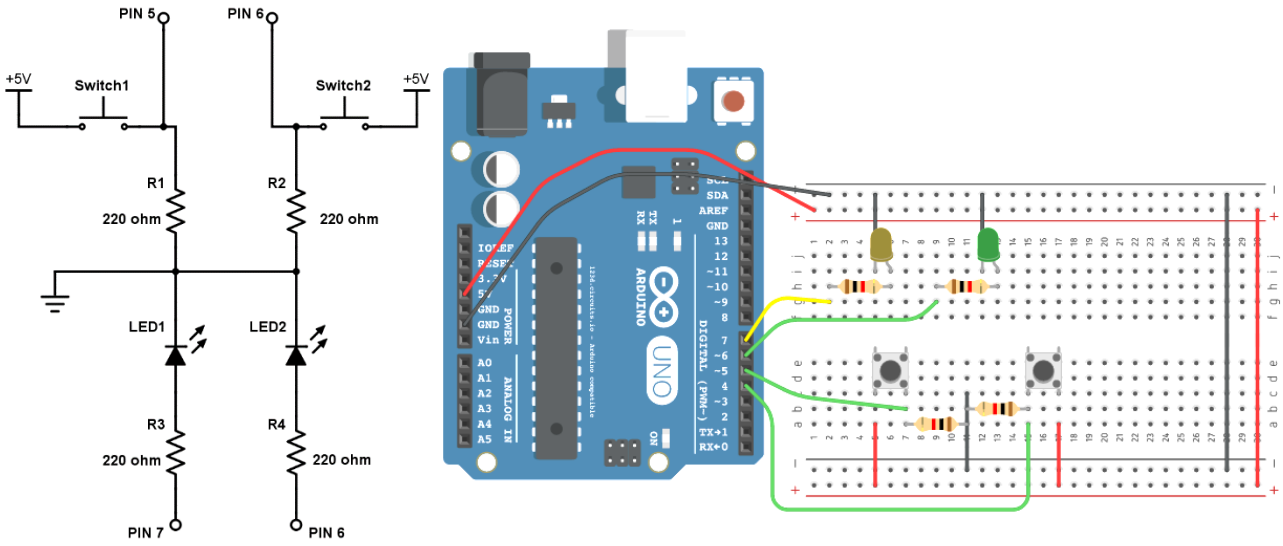

Figure 2: LEDs and switches example

Write your observations in the area below.

2.2 An example – Analog Read

In this example we will discuss how to read an analog signal using Arduino Uno. Unlike, digital signal which has a high and low state, analog signal has time-dependent continuous voltage change.

In order to read the analog value the Arduino board contains a 6 channel (8 channels on the Mini and Nano, 16 on the Mega), 10-bit analog to digital converter. ***This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023***. This yields a resolution between readings of: 5 volts / 1024 units or, .0049 volts (4.9 mV) per unit.

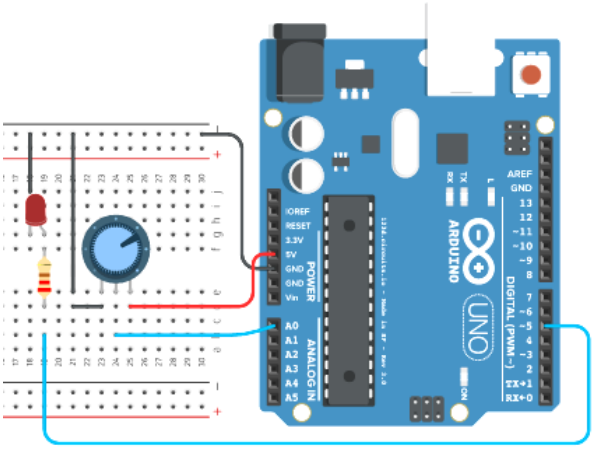

Figure 3: Voltage measurement using Analog Read

You need to construct the circuit shown in figure 3 and download the code (Analogread.ino) from Moodle. Please go through the program and analyse the functionality of the program. Check the values being displayed on the Serial Monitor and note the changes when you increase or decrease the resistance using Potentiometer.

Write your observations in the area below.

2.3 An example – LCD Connection

In this example we will discuss how to connect a LCD with Arduino UNO. LCDs and other displays greatly enrich the man-machine interaction. When a LCD is connected to a controller it requires multiple input/output (IOs). To solve this problem, the provided LCD comes with an additional attached module (I2C LCD 2004). LCD 2004 module has 4 pins which needs to be connected with the below mentioned (Table 1) pins of Arduino.

S. No	LCD 2004	Connection with Arduino UNO
1	GND	GND
2	VCC	5V
3	SDA	A4 pin
4	SCL	A5 pin

Table 1: LCD Connection Pins

Figure 4: LCD front and back view

The LCD provided has 20 Columns and 4 Rows. After connecting the LCD to the Arduino Uno you need to add the LCD library. Please download LiquidCrystal_I2C.zip file from the Moodle and save it in your project folder. Open the Arduino IDE, Select Sketch -> Include Library -> Add ZIP Library.

Download the LCD program (LCD_HelloWorld.ino) from Moodle and observe the output. Define what is the output you see on each line of LCD.

Write your observations in the area below.

Section 3: Solve the Challenges

You need to write program and build circuit for the given challenges and show the result to Lab Demonstrators and get your lab report signed.

All of section 3 will be a single program file.

3.1 Display Temperature on LCD's 1st Row

Use LM35 temperature IC and connect it to the breadboard and Arduino Board. LM35 will provide you the variable voltage depending upon the temperature. Using analog to digital conversion, read the voltage and then convert it to temperature in Celsius. Display the temperature value on the 2nd Row of LCD and keep updating.

Note: Take room temperature as 23ºC

3.2 Display Temperature and Group Names.

Write name of your group member's in charter string as shown below,

```
char* myStrings[]={"Name1", "Name 2", "Name 3", "Name 4" };
```

Now, attach Two Push buttons on your breadboard and to Arduino Uno. As LCDs is already attached and it should be displaying temperature in the 2nd row.

On the 4th Row of LCD your group members ***Name1*** should appear. Next ***name*** should appear if Button 2 is pressed, previous ***name*** should appear if Button 1 is pressed.

Your final output on LCD should look like this: -

Print given text on the 1st line {" **LAB-1** "}

Print temperature on the 2nd line {" **Temp = 23 'C** "} // keep updating temperature

Print given text on the 3rd line {"**Group Members**"}

On the 4th line print name of your group member {"**Name1**"} that will change depending on which button is pressed.

LAB-1 END