Principles of Measurement & Instrumentation I

Laboratory

PHYS417

Laboratory Report

Abdullah Burkan BEREKETOĞLU

Experiment 2- Arduino-Based Measurement Devices

Submitted By: Abdullah Burkan BEREKETOĞLU

Submitted to: Deniz Orhun BOZTEMUR & Enver BULUR

Student Id: 2355170

Experiment Date: 26.11.2019

Submission Date: 9.12.2019

OBJECTIVES

Our main purpose in the experiment is to learn microcontrollers, microdevices, especially Arduino, how it works and why we are using these types of devices. In achieving the main purpose, in the procedure/process we learned about basic principle of the PWM(Pulse Width Modulation), and about difference between micro - processor/controller/computer, In the Arduino UNO, we learned some function in it's IDE, and controlled analog signals by digital responses from UNO.

INTRODUCTION

Microcontrollers

Microcontroller is a micro device that controls other devices and machinery, since it is small and controller, we call it microcontroller. In the microcontroller, we contain a processor and a memory similar to what we see in a computer, but this one can solve small and specific tasks. Arduino UNO is an example of a microcontroller for the experiment.

Figure 1: Picture of Microcontroller

Microprocessor

Microprocessors basically the processing part of the system. As in a factory, you can crystallize, evaporate, boil, mix, these can be thought as chemical processes, and in the logic of process what we are doing is actually in theory the same. A process is a system that changes the information on a way that process is given. For example 2 +2. We have two of 2's and the process is + which is addition. This is what microprocessor does. It takes instructions and other logical operations and perform them with its ALU (Arithmetic and Logical Unit). It's register acts as quick access memory for the information process and it's control unit controls the flow which can be thought as pipeline of a system.

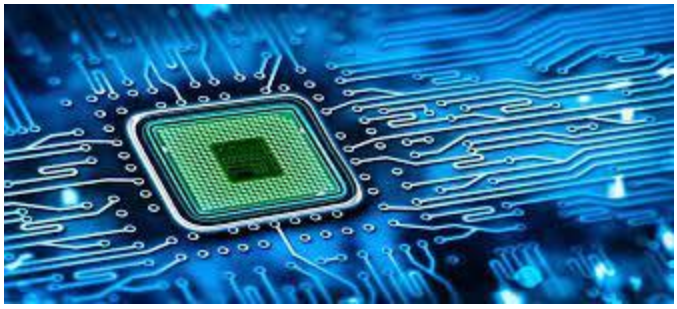

Figure 2: Picture of Microprocessor

Microcomputer

Microcomputers can be thought as computers with minimum amount of microprocessors, program memories, data memories, and I/O' which are designed for specific tasks, hence minimum amount they are small. Cheap and handy. They might also have timers, ADC's, etc. Also they might have timers, ADC etc…

Figure 3: Picture of Microcomputer

Basic Principle of PWM(Pulse Width Modulation)

PWM balances the nature of analog signal, which has infinite resolution in time, by this method we can control it directly. We control it by digital sources. E.g changing duty cycles, we can change width modulation so we can take more or less voltage.

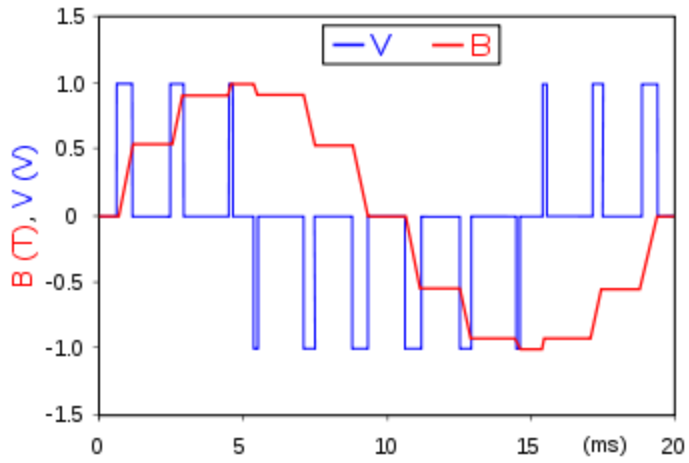

Figure 4: Picture of Pulse Width Modulation

Arduino

Arduino is an open hardware board setup for designing devices. They are easy to handle, can read inputs, light sensor, send message, and can turn responses into output. On the board, there is microcontroller which takes the instructions for the process which is delivered by the Arduino software program. We used Arduino UNO, and its pins, digital/ analog I/O and so.

Figure 5: Picture of Arduino Uno

EQUIPMENT

Breadboard

Arduino UNO

Resistors

LED

Push Button

Potentiometer

Procedure and Calculations

Blink LED

1- In the system, I used a blink code to make a blink in the LED for one second, used a current-limiting resistor to prevent the LED to burn. In the setup, I used pin 11 for resistor output, connected a long leg of the LED to this resistor, grounded the other leg on the Arduino ground.

Program to Blink LED

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1024);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(200);                        // wait for  0.25 seconds
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(512);                        // wait for 0.5 seconds
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
```

```
delay(200);                     // wait for  0.25 seconds
digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(512);                     // wait for 0.5 seconds
digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
delay(200);                     // wait for  0.25 seconds
digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(512);                     // wait for 0.5 seconds
digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
delay(200);
digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(1024);                     // wait for a second
digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
delay(200);                    // wait for  0.25 seconds
digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(1024);                     // wait for a second
digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
delay(200);                    // wait for  0.25 seconds
digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(1024);                     // wait for a second
digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
delay(200);                     // wait for  0.25 seconds
digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(1024);                     // wait for a second
digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
```

```
  delay(256);                    // wait for  0.25 seconds
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1024);                    // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(256);                     // wait for  0.25 seconds
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(512);                    // wait for 0.5 seconds
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(256);                     // wait for  0.25 seconds
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(512);                    // wait for 0.5 seconds
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(256);                    // wait for  0.25 seconds
}
```

Program to Blink LED without delay Function

```
// constants won't change. Used here to set a pin number:
const int ledPin =  LED_BUILTIN;// the number of the LED pin

// Variables will change:
int ledState = LOW;             // ledState used to set the LED

// Generally, you should use "unsigned long" for variables that hold time
```

```
// The value will quickly become too large for an int (integer) to store
unsigned long previousMillis = 0;        // will store last time LED was updated

// constants won't change:
const long interval = 1000;           // interval at which to blink (milliseconds)

void setup() {
  // set the digital pin as output:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // here is where you'd put code that needs to be running all the time.

  // check to see if it's time to blink the LED; that is, if the difference
  // between the current time and last time you blinked the LED is bigger than
  // the interval at which you want to blink the LED.
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;

    // if the LED is off turn it on and vice-versa:
```

```
    if (ledState == LOW) {
      ledState = HIGH;
    } else {
      ledState = LOW;
    }

    // set the LED with the ledState of the variable:
    digitalWrite(ledPin, ledState);
  }
}
```

Read Output of Potentiometer

2- In this part used a 1k ohm resistor, connected one terminal to 5V, and the other to the ground, later I connect the output of the potentiometer to A1. I read the output by using the program given below.

Program to Read Output of Potentiometer

```
void setup() {
  //It set up Arduino to run.

  Serial.begin(9600);
  // It adjusts rate of data in bits per second for transmission.
}

void loop() {
  //It makes our operations to run again and again.
```

```
  int val = analogRead(A1);
  //I determined a newvalue and I used map function to adjust numbers to a new range.
  delay(100);
  // It pauses the program for specified time.
    Serial.println(val);
  // It prints the values on serial port screen, so we can see it.
}
```

Mapping the results

We mapped the result to 0 to 5 V. From 0 to 1023.

Program to Read Results in Scale 0 to 5

```
void setup() {
  //It set up Arduino to run.

  Serial.begin(9600);
  // It adjusts rate of data in bits per second for transmission.
}

void loop() {
  //It makes our operations to run again and again.

  int val = analogRead(A1);
  //It reads A1 as an analog input. "int" means integer.
  int newVal = map(val,0,1023,0,5);
  //I determined a newvalue and I used map function to adjust numbers to a new range.
```

```
  delay(100);
  // It pauses the program for specified time.
    Serial.println(newVal);
  // It prints the values on serial port screen, so we can see it.
}
```

In results, I saw 0 to 1023, which is what is expected in the system the resolution is 10 bits which equal, 2^10 = 1024 from 0 to 1023. Deniz hoca mentioned that we should see 1024 though or something like that but I couldn't remember it well I definitely remember that for some time Voltage was not showing 5, but, we fixed it by changing the val and using newVal.

Control the Brightness of an LED

3- In this part, I combined the potentiometer and LED by using the same circuits of part 1 and 2. Controlled the brightness by changing the resistance of the potentiometer by turning it..

Program to Control the Brighness of an LED

```
const int analog_Pin = A1;
//It makes behavior of the analogpin as constant.
const int led_Pin = 11;
void setup() {
  //It set up Arduino to run.
  pinMode(led_Pin,OUTPUT);
  // It set the specified pin as a INPUT or OUTPUT.
  Serial.begin(9600);
  // It adjusts rate of data in bits per second for transmission.
}

void loop() {
```

```
  // It makes our operations to run again and again.

  int analog_Value = analogRead(analog_Pin);

  // It reads analogPin as an analog input. "int" means integer. "analogValue" is a thing we determined.

  int brightness = map(analog_Value, 0, 1023, 0 ,255);

  //the range of brightness  from 0 to 255 by using map function.

  //0-255 corresponds to %100 duty cyle.

  analogWrite(led_Pin,brightness);

  //It writes analog value to ledPin.It is based on the PWM technique which is used for getting analog results from digital sources.

  delay(100);

  // It pauses the program for specified time.

}
```

Count the Number of Clicks on a Push Button

4- Used a button, and a resistor in this part. I made a relation pin 2 to the button, button to the resistor, resistor to the ground, and 5V to the button. I designed a system to count the numbers of clicks by using the Arduino interrupt function . When clicked I saw, increase in the value at the serial port screen. When I clicked to button, I saw that the value increased on the serial port screen.

Program to Count the Number of Clicks on a Push Button

```
const byte interruptPin = 2;

volatile byte state = 0;


void setup() {

  pinMode(interruptPin, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(interruptPin), flag, FALLING);

  Serial.begin(9600);

}
```

```
void loop() {
  Serial.println(state);
  delay(50);
}


void flag() {
  state++;
}
```

I first used interrupt pin as 3 which is not reserved for interrupt hence we couldn't achieve the interrupt function. Other than that I couldn't achieve the number of clicks accurately with low resistances, due to the time constant of the capacitor. This is an RC current T= RC. Voltage lags current in capacitive circuits. Hence more resistive system will prevent the sensor to measure oscillations in the button to be counted.

Count the Number of Clicks on a Push Button (Debounce Circuit)

5- In the part 5, what we did was actually what solved the Push button click issue. As I mentioned earlier in part 4 we solved the mechanic oscillation problem with the capacitor due to its time constant with a high resistor, hence getting the current lead in the system approximately for keyboards and so 1-2 seconds. Also as I said due to mechanic oscillation of the spring of the button when we click to the button. Capacitor as mentioned above solves by the current lead in the system hence, current decay is slower and oscillations above the threshold are eliminated.

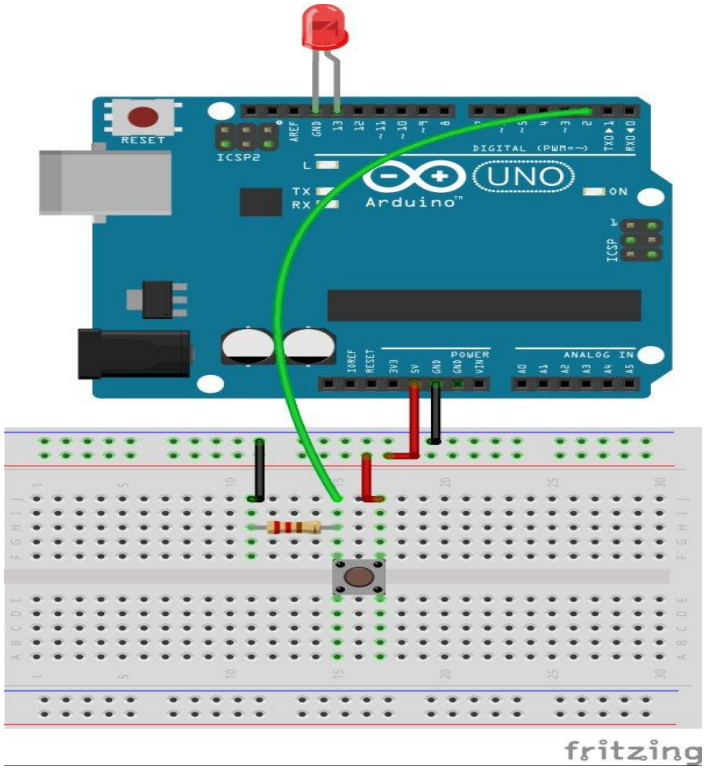

Figure 6: Picture of RC Debouncing Circuit

In this part, the same code as used in the step 4 is used since it is just the system with a capacitor in it. This problem could be also solved with determining a software threshold, but we solved it by using a capacitor instead (hardware). This is called RC debouncing. If the Resistance is low or we can say that RC is low then the noises may still interrupt, and give more clicks then clicked. There is also another issue, if the RC is higher than the click period in the system, then there may be no output for some clicks. For example, if T = 2s and we click in periods of 1s. In every 2 seconds, 1 click is not counted,

DISCUSSION

I was really nervous before using Arduino, and in any other parts of these experiments, because I don't have high self-esteem on the case. Because I was using Arduino for the first time in my life, I thought that it was going to be really hard and complicated. In the preliminary part, I was really scared that we needed to know device programming beforehand, but that was not the case. We had an IDE that had the demos so we didn't really need something like OOP C++ Qt for software engineering, after I did the experiment and found the codes, I was really relieved. It gave me a lot self-esteem on the case.

REFERENCES

Arduino Button Debounce Tutorial - (duino4projects.com)

Alışverişe Devam Edin (samm.com)

PWM: Pulse Width Modulation: What is it and how does it work? (analogictips.com)

Microcontroller Vs Microprocessor | HP® Tech Takes

What is a Microcontroller? Programming, Definition, Types & Examples - The Engineering Projects

Microcomputer - Wikipedia