ELEC1601 Exercise 1 Lab Report

SID: 530328265, Lab Group: T18-18

***Abstract*—In this report, we will go over what we have completed in the past two lab sessions, explain each step as well as discuss setbacks, lessons and possible real world applications encountered during the two lab sessions.**

I. Introduction

THE first lab session introduced Arduino and basic circuits for tasks like blinking LED lights controlled through buttons. To prepare, I read through the outline then experimented with the exercises on my own, giving me questions that I wanted to ask beforehand so I can focus my time on important subject material. During the lab, I asked my tutors the questions that I prepared and the parts that I struggled with. After that, I practiced after the lab to revise. At the first lab, although my team only had three members, our team succeeded in completing every exercise. In week 3, with a new member, it was easier for my team to pass every exercise so we finished 15 minutes early. In our final implementation, we succeed in creating real circuits that meet the requirements of the exercise 1.

II. Background

A. Tools(TinkerCAD/ Arduino IDE)

TinkerCAD: a digital and analog electronics simulator [1]. The first step in engineering a new concept is often to simulate your design before physically constructing it. Simulation also allows you to safely explore how your design performs with great freedom and focus on your coding. [2].

Arduino IDE: an open-sourse software and mainly used for writting, compiling and uploading code to almost all Arduino Modules. It is an official Arduino software, making code compilation easy enough for a beginner [3].

B. Computer (Arduino UNO R3)

Arduino UNO is a microcontroller board with 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or a power source to get started [4].

C. Breadboard

A breadboard: a rectangular board with many holes in it, which let you easily insert electronic components to prototype an electronic circuit [5].

D. LED (output)

A LED is a semiconductive device that emits light when current flows through it. The longer lead is the anode (positive side) and the other, negative side is the ”cathode.”. The brightness of LED is dependent on the current [6].

E. Buttons (input)

A button is a mechanical device used to control an electrical circuit in which the operator manually presses a button to actuate an internal switching mechanism [7].

F. Resistors and wires

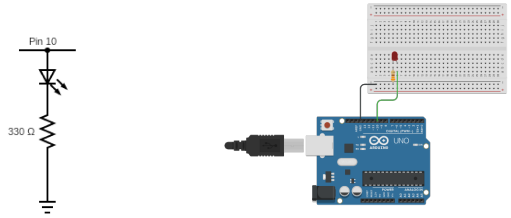
A resistor: is an electrical component that regulates the flow of electrical current to a desired value in an electronic circuit [8]. A summary table of all components is included.

	LED	***Resistors***	***Breadboard***	***Arduino***	***buttons***	***Wires***
Number	4	4(330Ω), 2(10000Ω)	1	1	2	various

III. Method

A. Part I: Blinking an LED

Firstly, we saw the model circuit and have to understand each component and its purpose. To prevent high current running through the LED, we used a 330Ω resistor, a middle value enough to protect the LED and keep an adequate current. We used a LED to give a physical indication of the signal from pin 10. When there's a signal, the LED will light up and it will turn off when there isn't. The algorithm 1 will make it so that the LED blinks on and off every second. To make the LED blink twice as fast, we halved the delay(1000ms to 500ms).



(a) Circuit Diagram (b) Diagram of connections

Algorithm 1 Pseudocode for Part 1 (first part)

```
Assign LED to an output pin 10
while True do
    Make LED output pin high
    Delay for 1000ms
    Make LED output low
    Delay for 1000ms
end while
```

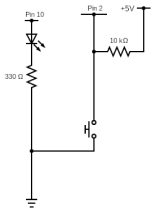
Algorithm 2 Pseudocode for Part 1 (second part)

```
Set serial communication speed to 9600 baud
while True do
    Print a new "Hello, World!" line to the serial port
    Delay for 1000ms
end while
```

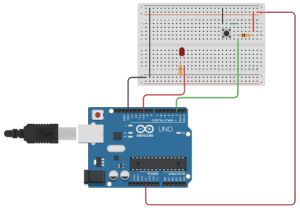
This algorithm 2 will print a new line of "Hello, World!" every second. To print the line "Hello, Group 18", we changed the value of the string. To print "How are you finding today's lab?" to a new line, we added a Serial.println() function. Serial.print() prints out on the same line, similar to Serial.println() but without creating a new line. \n creates a new string. We deduced that Serial.println() is like Serial.print() but with an \n added internally at the end.

B. Part II

The circuit can be splitted into two parts: the button and the LED. If the button is not pressed, there is a connection between 5V and Pin 2(the PinMode declared in void setup) through a resistor. Thus, this connection means Pin2 will have the value of 5V. Otherwise, Pin 2 is connected to ground through the switch, and 5V through a resistor. Since there is less resistance in the switch than the resistor, Pin2 is essentially connected to GND, so has a voltage of 0V. Thus, Pin2 will have a different signal 1(5V) or 0(0V), which will send the signal to the Arduino. The LED part is similar to the circuit of part 1.



(a) Circuit Diagram



(b) Diagram of connections

Algorithm 3 Pseudocode for Part2

```
Declare byte variable prevState
Declare byte variable prevLedState and assign it the value 0
Assign button to input pin 2
Assign the LED to output pin 13
Set serial communication speed to 9600 baud
Assign prevState the value of pin 2
Assign pin 13 the value of prevLedState (0)
while True do
    Assign the variable "button" the value of pin 2
    Print a new line of the value of the "button" variable
    if button's value is not equal to prevState's value then
        Assign prevState the value of button variable
        Negate the value of prevLedState
        Assign pin 13 the value of prevLedState
    end if
end while
```

We declared the two variables, one for the starting button's state and one for the previous state of the LED (starting with 0). In the while loop, we use the button variable to store the button's state and print it. If the values of the button variable and the previous state variable aren't equal, we save the button variable's value to prevState and change the state of the LED output pin.

C. Part III (The same circuit as part 2.)

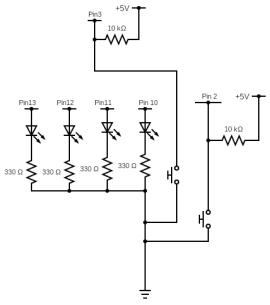
Algorithm 4 Pseudocode for Part 3

```
count ← 0
Assign pin 2 as an input pin
Assign pin 10 as an output pin
Set the communication speed to 9600 baud
attach interrupt to pin 2, calling changeLED when FALLING is detected
while True do
    if count is odd then
        Assign pin 10 the value of HIGH
    else
        Assign pin 10 the value of LOW
    end if
end while
procedure CHANGELED
    Add 1 to the count variable when called
end procedure
```

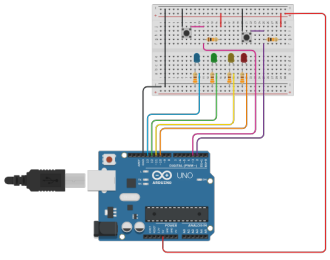
First, we declare a button variable for its pin and a count variable, set to 0.We declare input pin for the button and output pin for the LED. After that, we create an interrupt service routine (ISR) using the button input pin. Everytime the ISR is called, the count variable gets increased by 1 and if the count variable is odd, the light will turn on, turning off otherwise.

The ISR is like attaching a door bell to notify you, which is more efficient than polling (constantly checking). When the interrupt event happens, the microcontroller will run ISR and then continue the while loop that will react to the ISR response. Keeping the ISR short is important to avoid unexpected behaviors when two interrupts happen at the same time.

D. Part IV



(a) Circuit Diagram



(b) Diagram of connections

Algorithm 5 Pseudocode for Part 4

```
count ← 0
Assign pin 10, 11, 12, 13 as output pins
Assign pins 2 and 3 as input pins
Set communication speed to 9600 baud
attach interrupt to pin 2, calling incrementCount when FALLING is detected
attach interrupt to pin 3, calling reset when FALLING is detected
while True do
    if count equals to 0 then
        Set pin 10 to 13 outputs to LOW
    else if count equals to 1 then
        Set pin 10 output to HIGH
    else if count equals to 2 then
        Set pin 11 output to HIGH
    else if count equals to 3 then
        count equals to HIGH
    else if count equals to 4 then
        count equals to HIGH
    else if count more than 4 then
        Set pin 10, 11, 12, 13 output to HIGH
        Delay for 1000ms
        Set pin 10, 11, 12, 13 output to LOW
        Delay for 1000ms
    end if
end while
procedure INCREMENTCOUNT
    Adding one to the count variable
    Print out the count variable to the screen when called
end procedure
procedure RESET
    Assigning the count variable to the value of 0
end procedure
```

For part 4, we can split the circuit into three parts: the LEDs, the turn on button and the shutdown button. The LEDs part is connected in a way similar to part 3, adding three new LEDs, using output pin 11, 12, 13 for each new LED. The turn on button is connected similarly to part 2. The shutdown button is connected similarly to part 2 but uses pin 3 instead.

The code in algorithm 5 first declares the variable count and the two interrupt pins. After that, we set pins 10 to 13 as output pins, set up the 2 interrupt pins and set the communication speed. We declare two ISRs, one concerning pin 0, which calls the incrementCount function which adds to the count variable and prints it and the other calls the reset function which set the count variable to 0. In the while loop, we regulate the LEDs. For count value is more or equal to 5, it will light up the same number of LEDs (starting from the pin 10's LED to the pin 13's). The 4 LEDs will blink every half a second. When the reset function is called, the count value is set to 0 and turns off all the LEDs.

IV. Results

We successfully completed all 4 activities in both simulation and real conditions despite some setbacks (signal noise, inexperience in circuitry and coding). We witnessed signal noise in activity 3 when the ISR response is sometimes delayed despite working perfectly in the simulation. After rechecking, We and our lab demonstrators attributed it to signal noise. We couldn't fix the issue but a resistor-capacitor (RC) circuit could smooth out the signal noise. That's a current limitation of our circuit. Reading resistors' values is particularly difficult before we get used to it. During the second task, we lost another 20 minutes figuring out why our circuit was not working which was due to a misplaced button. Through the experience, we have gotten more experienced at reading resistors and checking the circuit. We also experienced multiple coding bugs due to our lack of experience with C++. By carefully reading our comment and trying to isolate the issue, we have become much more competent at detecting and fixing bugs.

V. Relation to real-world electronics

The coding and circuit we completed are fundamental to many real world applications. The concepts involved in part 1, 2 (resistors to regulate current, setting up input/output channels) are used in every circuit. The algorithm and hardware of exercise 3, 4 can be utilized in the design of a Christmas light. The ISR can be activated by the same button mechanism in exercise 3 and it can use a counter variable to decide the behaviors of the LEDs.

- The pseudocode of the light system is similar to activity 4 but with more responses for each counter variable's value (count = 1 lights on, 2 lights blink, 3 lights blink faster, 4 reset).
- There are many problems to overcome. The effect of signal noise across a large system may be more significant, regulating the current and maintaining reliability (what if a light burns out?) will be more difficult. A power source enough to power it may also pose safety hazards.

VI. Conclusion

The two lab sessions have been a success and taught us important experience in both simulated and physical circuitry, programming's applications, team work, how to write professional reports in LaTeX and how to prepare for future lab sessions.

References

[1] Maker.io Staff, "Getting Started with Tinkercad Circuits." https://www.digikey.com/en/maker/blogs/2022/getting-started-with-tinkercad-circuits, 4 2022.

[2] The University of Sydney, "Weeks 2 and 3: Lab Exercise 1 - Intro, LEDs, Interrupts." https://canvas.sydney.edu.au/courses/52848/pages/weeks-2-and-3-lab-exercise-1-intro-leds-interrupts?module_item_id=2053576.

[3] David watson, "Introduction to ArduinoIDE." https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide.html, 10 2018.

[4] Arduino Company, "Arduino Uno R3." https://docs.arduino.cc/hardware/uno-rev3.

[5] science buddies, "How to Use a Breadboard for Electronics and Circuits." https://www.sciencebuddies.org/science-fair-projects/references/how-to-use-a-breadboard#more-about.

[6] Wikipedia, "Light-emitting diode." https://en.wikipedia.org/wiki/Light-emitting_diode.

[7] Ryan Smoot, "ush Button Switches 101." https://www.cuidevices.com/blog/push-button-switches-101.

[8] TechTarget Contributor, "resistor." https://www.techtarget.com/whatis/definition/resistor.