Lab 1 Report

Gang Hyun Kim, 40097242

September 21, 2021

1. The purpose of this lab is to introduce students to some basics of embedded systems through the Arduino nano microchip. This lab allows students to understand the basic hardware implications as well as the software components used when developing embedded systems.
2. For this lab, there are a few hardware resources that are required which are: arduino microchip, breadboard, wires, resistors, LED array, computer/laptop and a mini-usb to usb converter.

For all tasks, the Arduino microchip must be placed on the breadboard. The mini-usb to usb converter must be plugged in to a computer for the experiments to be successful.

For task 1, a wire must be placed going from the ground of the digital side of the microchip to digital pin 6.

Task 2 builds on top of the circuit of task 1. Place an LED array next to a resistor. Add a new wire going from digital pin 9 to an LED on the LED array. Add another wire going from the ground to the opposite end of the LED array which should allow the LED on the array to be powered.

Task 3 does not require the same circuit as task 1 and 2. For task 3, a wire must be placed going from the ground of the digital side of the microchip to the ground of an LED on the LED array. Then, connect a wire from the digital pin 9 to the grounded LED which should allow the LED to be powered.

Task 4 and 5 does not require any changes from the circuit of task 3.

1. For this lab, Arduino IDE is recommended to be used. MinGW compiler is needed to compile the C language code which is used to write the software to control the Arduino. If the computer does not support USB 2.0 serial, downloading CH340 driver will fix the issue. Once the Arduino IDE is downloaded and the microchip is connected to the computer, it is important to make some adjustments to the configurations of the IDE. In the Tools option in the menu, change the board from Arduino Uno to Arduino Nano, then in the same Tools option, change the Processor to ATmega328P (Old Bootloader).

If the Arduino Nano is properly plugged into the computer, the microchip should have a red light turned on. Furthermore, in the Tools option of the IDE, the Port option should be available. In this case, select the port that the Arduino is connected to. Once this is done, on the top right corner, there is a button for the serial monitor. Click on the button and check if the monitor is set to a baud of 9600 and change the option of the left of the baud settings from Newline to No line ending. Now, everything should be ready to go.

1. Task1: The goal of this task is to be able to reproduce the logical inverter with the given resources. A logical inverter takes an input and inverts its value as an output. Therefore, to be able to complete this task, one must first understand how to get an input into the microchip.

Text

Description automatically generated

The pinMode of pin 6 needs to be set to INPUT\_PULLUP because this makes it so that when an input pin is grounded, there is a signal. This is important because an input is required, since the microchip is not attached to any other chips, the ground needs to become the “source” of input pin. The LED\_BUILTIN is a special LED that is integrated in the microchip. By writing to this LED and by putting a not sign in front of it, the logical inverter is implemented. To read the information from the pin however, digitalRead method needs to be implemented. Therefore, the full implementation should look something similar to digitalWrite(LED\_BUILTIN, !digitalRead(pin\_of\_choice). The special LED will only light up when the inverted pin 6 is connected to the ground which is the logical false.

Task2: The goal of this task is to be able to turn on and off an LED. This task’s main challenges are in the hardware, however a bit of modification needs to be made in the code as well.

Text, letter

Description automatically generated

The code for this task is the exact same as the first task with a minor difference. The LED\_BUILTIN is replaced by a different pin which should be connected to an LED on the LED array. In the setup method, the new pin needs to switch its mode by adding pinMode(pin\_of\_choice, OUTPUT). By connecting and disconnecting the wire connected to the input pin, it is possible to turn on and off the LED.

Task3: The goal of this task is to be able to cycle through different levels of brightness of an LED.

Graphical user interface, text, application, email

Description automatically generated

This task is different from the previous two. Only one pin is needed for this experiment. By setting the pinMode of the selected pin to OUTPUT, the pin should be ready to go. By doing analogWrite, it is possible to have fine control over the brightness of the selected pin by giving it a value between the ranges of 255 and 0 where the value of 255 is the brightest. By running a for loop, it is possible to run through all levels of brightness of the LED.

Task4: The goal of this task is to be able to print out the brightness of the output pin into the serial monitor.

Graphical user interface, text, application, email

Description automatically generated

The code for task4 is almost the same as task3 with minor differences. The serial monitor is required to be able to digitally observe the brightness level of a pin. In the setup method, Serial.begin method needs to be implemented. The begin method takes in the amount of baud the microchip sends out, so the implementation should look like Serial.begin(9600). It is possible to print out the brightness of the pin by adding Serial.print or Serial.println. The pin’s brightness is accessed by analogRead method. Therefore, by implementing Serial.println(analogRead(pin\_of\_choice)), it is possible to observe the brightness in the serial monitor.

Task5: The goal of this last task is to be able to receive commands from the serial monitor and adjust the brightness according to the input.

Text

Description automatically generated

This task does not require the observation of the level of brightness of the pin, therefore, analogRead method is not needed. This task needs to take an input from the serial monitor and change the pin’s brightness accordingly. To be able to do so, the code needs to check if a value has been input to the serial monitor. The Serial.available method is used to check this. This method returns the number of bytes that are available to read, so when it is greater than 0, it means that there has been an input from the user. The input value can be saved into a variable through the method Serial.parseInt which is one of many methods of doing so. By taking the variable value and inputting the value as the brightness of the selected pin through the analogWrite method, the LED will change its brightness to the input value in its next cycle.

1. For this lab, the Serial library was used to print information such as the brightness level and input values to modify the brightness level. The Arduino programming library was also used to be able to read/write digital pins as well as have fine control over the brightness through the analog read/write methods. It was also used to be able to set the pins on the Arduino to different modes such as input and output modes.
2. In conclusion, this lab was a through introduction to the Arduino library and embedded systems. I learned about how the hardware components interact with the software code. It also helped me refresh my memory on C programming language as well as taught me about how to use Serial and analog/digital read and write methods. The introduction to these basic methods opened up the possibility of many interesting projects in both the software level and on the hardware level.