Analog **Course: CSCI 3331**
Lab 4 **Section: 01**

Ahmed Muntasir Hossain

Description of the Lab

The purpose of the lab was to control the brightness of an LED using analog devices: temperature sensor, light sensor, and joystick. Each of the analog devices provide a voltage input from 0 to 5V to the Arduino, which is then converted to a 10-bit binary number by the Analog Digital Converter (ADC). The voltage input of the analog devices varies depending on the external input from the user. For instance, the analog input of the joystick is changed by moving it along the x and y-axis, the analog input of the temperature sensor is changed by heating or cooling the sensor, and the analog input of the light sensor is changed by exposing the sensor to different intensities of light. The ADC values of each sensor is converted to its corresponding pulse width modulation (PWM) value. This value is the duty cycle of a square wave which is generated in a particular Digital PWM pin determined by the programmer. The LED is connected to that pin which allows the user to vary its brightness by controlling the analog inputs.

The key characters ‘L’, ‘X’, ‘Y’, and ‘T’ have been preset for the light sensor, x-axis of the joystick, y-axis of the joystick, and temperature sensor, respectively. The user inputs the key character for the sensor which they intend to control using the serial monitor. The program uses by default the x-axis of the joystick as the sensor.

The raw ADC value or voltage that is produced by the analog devices is outputted to the serial monitor depending on the output type the user chooses. The key characters ‘R’ and ‘V’ have been preset for raw ADC value and voltage, respectively. The program indefinitely outputs the current readings of the chosen analog device. Lastly, the program outputs the raw ADC value by default if the user does not specify the type.

Analog Input Devices and their functionality:

> As the light intensity on the light sensor was increased, the raw ADC value decreased, and therefore the brightness of the LED decreased and vice versa. As the temperature of the temperature sensor was increased, the raw ADC value decreased, and therefore the brightness of the LED decreased and vice versa. For the joystick on a particular axis (x or y), the ADC value increased in one direction and decreased in the other. This caused the brightness of the LED to increase and decrease, respectively.

Power

The breadboard has two columns marked + (positive) and – (negative). One wire is connected to each column and it can be placed in any hole on that column. This is used to supply power from the Arduino to the breadboard. The positive and negative wires connected to the breadboard are connected to the 5V power pin and GND power pin on the Arduino, respectively.

Output

The output of this device is the RGB LED. It is an electrical component consisting of three LEDs Red, Blue, and Green. The RGB LED has 4 pins “R” (red), “G” (green), “B” (blue), and “–” (negative) which are connected parallelly on the breadboard (different rows). Since, only the Red LED is used for this lab, a single wire is connected to the “R” pin. The R wire is connected to

any of the digital PWM pins (pin number 3, 5, 6, 9, 10, 11) on the Arduino. The negative wire is connected to any hole (socket) on the negative line on the breadboard. This completes the wiring connection of the RGB LED. The R, G, and B pin markings on the LED may be incorrect and therefore to configure it correctly, the device should be tested. This should be done by setting each individual pin (LED color) to HIGH and observing its color. The color determined from the test can be used to know the correct configuration of the pins.

Input

The inputs of this device are the joystick, light sensor, temperature sensor and the Universal Asynchronous Receiver-Transmitter (UART) port.

The UART port on the Arduino is referred to as COM3 on my device. The name of the port may vary depending on the Arduino. The port is connected to a PC using a USB 2.0 cable (type A/B). Using the Serial Monitor, a message can be sent to the Arduino to be processed. Using the same port, a message can be transmitted to the PC to be displayed on the Serial Monitor.

The light sensor and temperature sensor has three pins “S”, “-” (negative) and the middle pin which are connected parallelly on the breadboard. One wire is connected to each pin serially. The middle pin wire is connected to the positive line, the negative pin wire is connected to the negative line, and the “S” wire is connected to any analog input pin (pin number A0 – A5) on the Arduino. This completes the wiring of both the sensors. Additionally, there may be an error in the markings of the pins. In both of my sensors the markings of S and the negative pin were swapped. To determine if there is an error, the serial monitor should be used to output the raw ADC or voltage value. If there is no change in the reading of the analog inputs when the temperature or light intensity is varied for the appropriate sensor, then there is an error in the markings and the wires should be swapped. Otherwise, the configuration is correct.

The joystick has 5 pins “GND”, “+5V”, “VRx”, “VRy”, and “SW” which are connected parallelly on the breadboard. For this lab, only the first fours pins were used because the SW pin is used to control the push button on the joystick. The push button was not used in this device. One wire is connected to each pin serially. The +5V pin wire is connected to the positive line, the GND pin wire is connected to the negative line, and the “VRx” and “VRy” pin wires are connected to any two analog input pins (pin number A0 – A5) on the Arduino. This completes the wiring of the joystick.

Parts of Arduino

I learned about analog input devices and pulse width modulation. An analog input device provides a voltage input between 0 to 5V to the Arduino depending on the external user input (moving the joystick, or varying temperature or light intensity). The voltage is converted to a 10-bit binary number using the ADC (Analog Digital Converter). The value of the ADC varies depending on the analog input and it is used to control an output device such as an LED or motor.

Pulse width modulation (PWM) is a process of producing “analog” results through digital means. A square wave (a signal which switches between on and off) is generated for a particular duty

cycle. The duty cycle is the percentage of time the signal is on for that wave's time period. As the signal continuously changes from on to off (5V to 0V), it produces a steady voltage between these two values therefore creating an analog output. To vary this output, the duty cycle is changed.

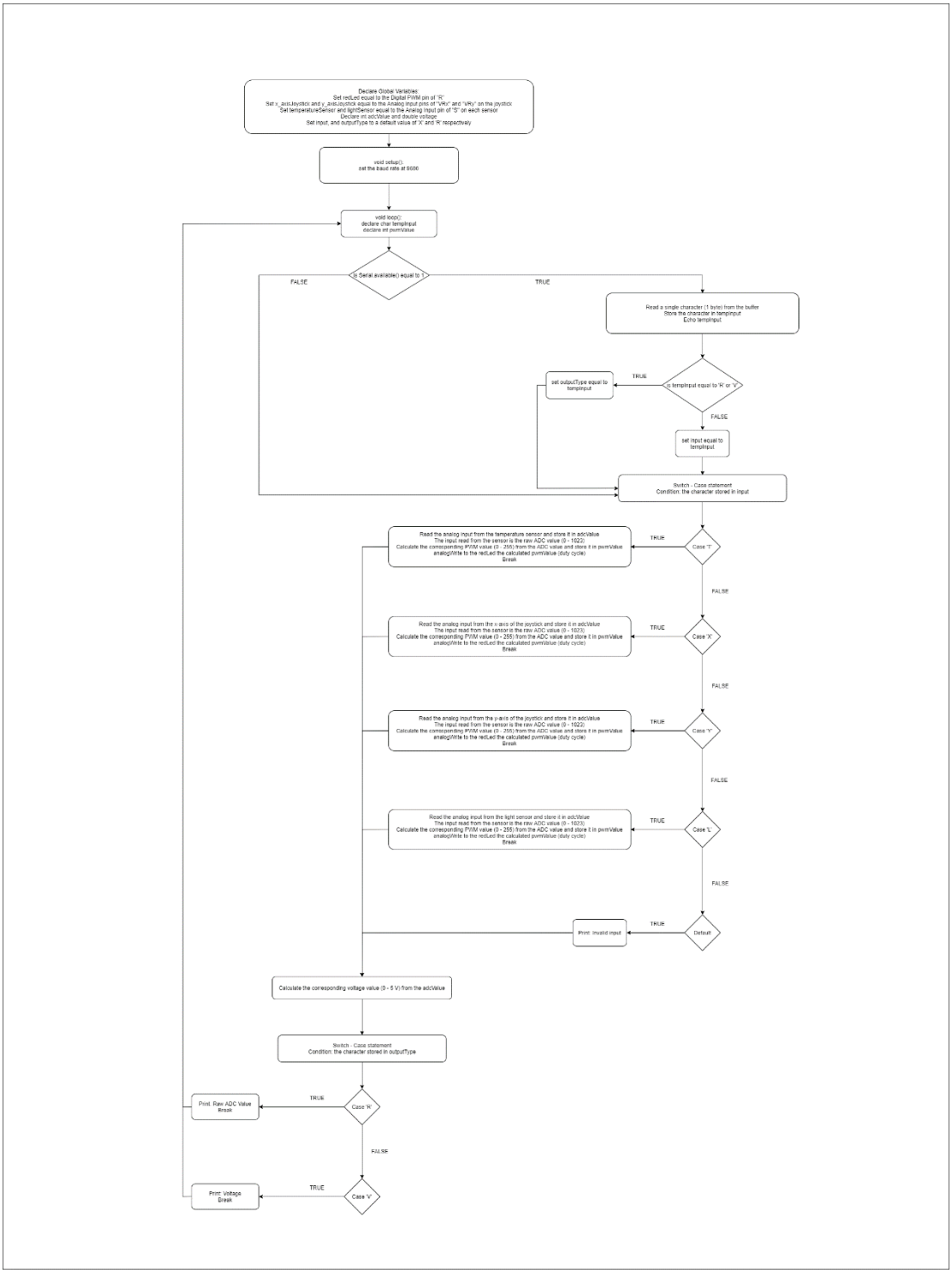
The output devices mentioned above are connected to digital PWM pins where such a wave would be generated. The duty cycle depends on the ADC readings received from the analog devices. The ADC readings are converted to duty cycle values by setting the highest ADC value to a 100% duty cycle and lowest ADC value to a 0% duty cycle to form a linear relationship. This relationship is used to calculate the duty cycle for any ADC reading. Therefore, to conclude, as the analog input varies, the ADC reading varies causing the duty cycle to change. This change in the pulse width (on time) creates a new steady voltage between 0 and 5V causing the output of the device (brightness of the LED, or speed of the motor) to vary.

Analog
Lab 4

Course: CSCI 3331
Section: 01

Ahmed Muntasir Hossain

Flow Chart of the Program Logic



Analog
Lab 4

Course: CSCI 3331
Section: 01

Ahmed Muntasir Hossain

Code

```
const int redLed = 5;            //Digital PWM pin for red LED

const int x_axisJoystick = A0;    //analog input pin for x-axis of the joystick
const int y_axisJoystick = A1;    //analog input pin for y-axis of the joystick
const int temperatureSensor = A2; //analog input pin for the temperature sensor
const int lightSensor = A3;       //analog input pin for the light sensor

int adcValue;
double voltage;

char input = 'X';       //default value of analog input device: x-axis of joystick
char outputType = 'R';  //default value of output type: raw ADC value

void setup() {

  Serial.begin(9600); //set baud rate
}

void loop() {

  char tempInput;   //stores input from user
  int pwmValue;     //stores the calculated duty cycle (PWM) from the raw ADC value

  if (Serial.available() == 1)  //if the buffer contains 1 character
  {
    tempInput = Serial.read();  //read the character and store it in tempInput
    Serial.println(tempInput);  //print the character

    if (tempInput == 'R' || tempInput == 'V') //set outputType equal to tempInput if tempInput contains the apporpriate key character ('R' or 'V')
    {
      outputType = tempInput;
    }

    else                        //else set input to tempInput
```

Ahmed Muntasir Hossain

```
  {
    input = tempInput;
  }
 }

 switch (input) //switch-case statement for analog input device
 {
  case 'T':                          //temperature sensor
    adcValue = analogRead(temperatureSensor);     //reads the analog input from sensor and stores the raw ADC value
    pwmValue = (255  * (adcValue - 123)) / 367;   //calculates the corresponding duty cycle for that raw ADC value
    analogWrite(redLed, pwmValue);              //writes to the output device (redLed) the calculated duty cycle to vary the brighness of the LED
    break;
  case 'X':                          //x-axis of joystick
    adcValue = analogRead(x_axisJoystick);
    pwmValue = 255 * adcValue / 1023;
    analogWrite(redLed, pwmValue);
    break;
  case 'Y':                          //y-axis of joystick
    adcValue = analogRead(y_axisJoystick);
    pwmValue = 255 * adcValue / 1023;
    analogWrite(redLed, pwmValue);
    break;
  case 'L':                          //light sensor
    adcValue = analogRead(lightSensor);
    pwmValue = 255 * adcValue / 1023 ;
    analogWrite(redLed, pwmValue);
    break;
  default:                           //invalid character inputted
    Serial.println("Invalid Input");
 }

 voltage = 5.0 * adcValue / 1023;  //calculate the corresponding voltage from the raw ADC value

 switch (outputType) //show appropriate output depending on the outputType chosen by the user
 {
  case 'R':
```

Ahmed Muntasir Hossain

```
    Serial.print("Raw ADC Value: ");   //print Raw ADC Value
    Serial.println(adcValue);
    break;
  case 'V':
    Serial.print("Voltage: ");        //print Voltage
    Serial.println(voltage);
    break;
 }

}
```