|  |  |  |
| --- | --- | --- |
| Logo, company name  Description automatically generated | **Lab #3** | **Controlling a Fan** |
|  |  |
| EECE.5520 Microprocessor II and Embedded System Design | |
| Name: Quoc Huy Huynh  Student ID: 01923052  Team name: Solo work | |
| Date submitted: 11/21/2021 | |
| Github page: <https://github.com/huyontheway/quochuy_huynh_eece5520_fall2021.git> | |
|  |  |

Demonstration method: recorded video

(<https://studentuml-my.sharepoint.com/:v:/g/personal/quochuy_huynh_student_uml_edu/ETIhJZAzwz5FlBZvNgj6C_cBClX8Z8IE3smUYQETQEzFZw?e=e1kcUn>

AND

<https://studentuml-my.sharepoint.com/:v:/g/personal/quochuy_huynh_student_uml_edu/ESLGJi1jEYZHqIuekE8HvgwB1rXeS0zdl3TEFA2IVVdO1w?e=F4rbPy>)

1. **Design**

*1.1 Hardware design*

A picture containing text, electronics

Description automatically generated

**Figure 1.** Drawing schematic of hardware design

* Table below is the list of all required parts of the lab.

|  |  |
| --- | --- |
| **Part** | **Quantity** |
| Arduino MEGA 2560 | 1 |
| DC motor | 1 |
| Power supply module | 1 |
| L293D | 1 |
| LCD display | 1 |
| DS1307 RTC | 1 |
| Push button | 1 |
| IR Receiver | 1 |
| IR Remote control | 1 |
| Jumper wires | As needed |

* DC motor:
  + To control the DC motor, a power supply is utilized. The DC motor is likely to use more power than what an Arduino board can supply. If we try to connect the motor directly to the board, there’s a high chance that it will damage the board.

A picture containing text, electronics

Description automatically generated

**Figure 2.** Power supply module [1]

* + Below are the product specifications of the power supply module: [1]

Text

Description automatically generated

* + An advantage of this module is that we can configure the left and right voltage independently. To select the voltages, we can move the short jumper cap to the appropriate voltage we desire.
  + One thing to note when placing the power supply module onto the breadboard is to correctly align (+) and (-) pin to the appropriate row on the breadboard.
  + To drive the motor, the L293D chip is utilized. It can control two motors independently. However, for our purpose only half of the pins on the L293D chip are used because we only have one DC motor to control.
  + Below are the product specifications of the L293D chip: [1]

Text, letter

Description automatically generated

* When the enable input is high, the associated drivers are enabled. The outputs are now active and in phase with inputs. On the other hand, when the enable input is low, those drivers are disabled. The outputs are now off and in high-impedance state which means the motor will be turned off.

Text

Description automatically generated

**Figure 3.** L293D pin layout [1]

* Pin 1 (M1 PWM) is connected to a PWM pin on the Arduino. In our case, it is connected to pin 5 on the Arduino. This pin directly controls the speed of the motor. The output can range from 0 to 255. 0 means that the motor is off, while at 255, the motor will spin at its max speed.
* M1 direction 0/1 and M1 direction 1/0 are connected to two digital pins on the Arduino. The two pins control the direction of the motor. One of them can be HIGH and the other can be LOW, the motor will spin in one direction. Reverse the two pins, the motor will spin in the opposite direction.
  + The push button is connected to pin number 3 as an input to detect if it is currently pressed. Whenever it is pushed, pin 3 will appear as HIGH and vice versa. The main purpose of this button is to reverse the current direction of the motor when it is pressed.
* LCD display:
  + The LCD display has LED backlight and can display two rows with 16 characters for each row.

A close-up of a computer

Description automatically generated with low confidence

**Figure 4.** LCD Display [1]

* Below is explanation about functionality of each pin of the LCD display: [1]

Graphical user interface, text

Description automatically generated

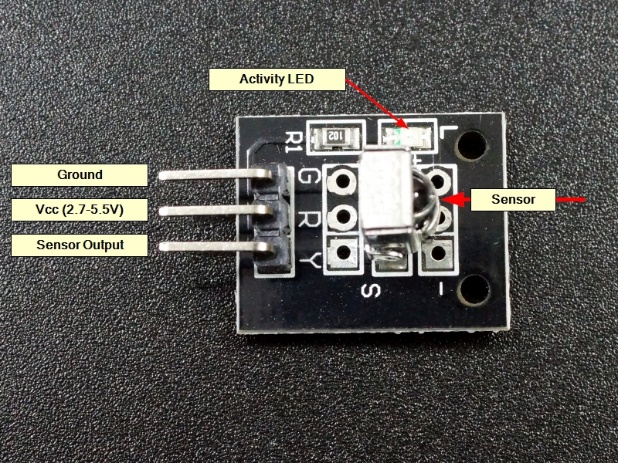
* + Six digital Arduino pins are used to control the LCD display. 5V and GND from the Arduino board are used to power the display.
  + A potentiometer is used to control the contrast of the display.
* RTC module:
  + The DS1307 provides seconds, minutes, hours, day, month and year information.

Graphical user interface

Description automatically generated

**Figure 5.** RTC module [1]

* + SCL pin is connect to pin 21 SCL on the Arduino. SDA pin is connected to pin 20 SDA on the Arduino.
  + The VCC pin plugs into the 5V port, and the GND plugs into the GND port.
* IR Receiver and IR Remote control:
  + IR Receiver is a microchip with a photocell that is used to listen to infrared light. An IR remote control has a matching IR LED which emits IR pulses to transfer data to the IR receiver to perform tasks in the system.
  + IR light is not visible to human eyes.



**Figure 6.** IR Receiver

* + The Y pin (data pin) on the IR receiver is connected to pin 2 of the Arduino. The G and R pins are connected GND and +5V on the Arduino.
  + When we press a button on the IR remote control, an IR pulse will be sent out and received by the IR receiver. This will then be converted to a hexadecimal number that we can utilize in our code to detect what button was being pressed.

*1.2 Software design*

1. Flow Diagram

Diagram

Description automatically generated

**Figure 7.** Flow diagram of the application

* At the beginning when the code starts, the DC motor will begin to spin at its maximum speed and in the clockwise direction. The program will listen to see whether the push button is pressed or any button on the IR remote is currently pressed to control the speed and direction of the motor.
* The information about the motor’s speed and direction will then be sent to the LCD to display.
* The RTC module will also configured when the system is on to retrieve real time information including date. Moth, year, hour, minute and second. This information will also be sent to the LCD to display and updated every one second.

1. Software program

* For the DC motor:
  + Pin 5 of the Arduino is connected to the ENABLE pin on the L293D chip which turns on and off the driver to control the motor.
  + Pin 3 and 4 of the Arduino control the direction the motor spins. Specifically, pin 3 controls DIRA port and pin 4 controls DIRB port. DIRA and DIRB can be either HIGH or LOW to control the direction of the motor.
  + *analogWrite(ENABLE,fanSpeed);*  will start the motor and set the spinning speed to fanSpeed that could be adjusted in the program.
  + The power supply module is supplied by a 9V battery.
* For the LCD display:
  + The library called LiquidCrystal is utilized to control the LCD display.
  + The library lets us configure what ports of the Arduino are connected to the LCD display through *LiquidCrystal lcd(7, 8, 9, 10, 11, 12)*. In addition, we need to initialize the library with the size of the display through *lcd.begin(16, 2)* which means 16 columns by 2 rows.
  + At this point we can start printing information on the display by using *lcd.print(“information”)*. Additionally, we can set the cursor on the display at a specific location by running *lcd.setCursor(column, row)*.
  + The information displayed on the LCD is updated every one second by a timer interrupt.
    - Timer1 interrupt is initialized and being used to generate pulse wave of frequency of 0.5Hz which means it can be utilized to update and display all the information we desire every one second.
* For the IR sensor:
* IRRemote.h library is utilized to operate the IR sensor and IR remote.
* When a button is pressed on the IR remote, an IR pulse will be sent to the IR receiver and converted to a hexadecimal number. Depending on the hexadecimal number, we can determine what button was pressed on the IR remote. An example is when the POWER button is pressed, the hexadecimal number converted is 0xFFA25D. The function *translateIR()* is in charge to translate this information to determine what button was pressed. Accordingly, we can perform a certain task with what button was pressed on the IR remote.
* For this lab, four buttons on the IR remote control are critical including FAST FORWARD, FAST BACK, UP and DOWN where FAST FORWARD controls the motor to spins in the clockwise direction, FAST BACK controls the motor to spin in the counterclockwise direction, UP to increase its speed and DOWN to decrease the speed.
* Push button
  + Main purpose of the push button is to reverse the direction of the motor. If the current direction in clockwise, it will reverse the motor direction to be counterclockwise and vice versa.
  + The integer variable called dir indicates what direction the motor is currently spinning. This is the crucial flag to control the direction of the motor when the button is pressed.
  + An int variable called buttonState is to indicate if the button is currently pressed.
* Real time clock module:
  + Wire.h and DS3231.h library are used to control the RTC module.
  + We can configure the module by sending compiling time to the Arduino by running *clock.setDateTime(\_\_DATE\_\_, \_\_TIME\_\_)*.
  + Inside the loop function of the code, we need to access the time data by running *dt = clock.getDateTime()*. After this point we can access the time info by using *dt.year*, *dt.month*, etc.
  + Note that we need to load the program from our computer every time we want to start over or after modifying the code to keep the time correctly. If we decide to press the reset button on the Arduino to restart the program, the time information will be incorrectly.

*1.3 Results*

The results of the lab worked as expected from the lab objectives. The demonstration video demonstrates all the functionality of the application. [2]

(1) use the Arduino to control the motor that turns a fan. The fan can run at different speeds

and in either direction (clockwise or counterclockwise)

(2) set up a Real Time Clock (RTC) and retrieve time information

(3) display the real-time clock and the fan’s rotation direction (“C” for clockwise, or “CC” or

counterclockwise) and speed (“Full”, “3/4”, “1/2”, or “0”) on a LCD

(4) update the information display every second

(5) allow pressing a push button to change the rotation direction of the fan

(6) use the IR receiver to receive and decode button press on an IR remote control.

(7) use the decoded information to switch the rotation direction and/or change the speed.

1. **Problems Encountered and Solved**

* Most of the parts for the lab, I have not worked with previously. At the beginning, I had to spend some time to get familiar with each of the part. The document from ELEGOO was very helpful to get started.
* Set up and organize breadboard for this lab was tricky at first because a lot of parts are being utilized.

1. **Personal contribution to the Lab**

* This is a solo lab, so I worked on all parts by myself.
  + Design the mechanism to control the DC motors to modify its speed and direction using a press button and IR remote and display this information on the LCD display.
  + Write the program to run on Arduino.
  + Wire up the circuits to make it work with the code.
  + Draw the schematics of the circuit and flow chart of the software.

1. **Lessons Learnt**

Throughout the lab, I had a chance to learn with IR receiver and RTC module which I have not had a chance to work with before this lab. The lab overall is very interesting because do we not only work with the parts, but we also have to retrieve information about each of them. For example, we have to be able to retrieve information about the DC motor such as its speed and direction. Doing it this way really helps me to understand not only about hardware but also about software part because I need to understand the program and the functionality of each part fully to be able to track what each part is doing in the system.

1. **References**

|  |  |
| --- | --- |
| [1] | ELEGOO, "The most complete starter kit tutorial for MEGA2560," ELEGOO. |
| [2] | Y. Luo, "Lab 3: Controlling a Fan," University of Massachusetts , Lowell, 2021. |