ECE110 Honor Final Report

Wakeup System

Jiarui Zou (jiaruiz3)

PS: This Honor project is done by Jiarui Zou, other two partners registered contribute to less than 5% of the project and dropped out of the Honor Section half way through the semester since they cannot get honor credit for the course.

Introduction:

Problem Description:

Get up in the morning has always been a life struggle for many students for their entire academic life, typically when they have to work on a due for 8 hours consecutively until 3Am in the morning and get up at 8 to go to the morning lecture. Indeed, an alarm clock can wake us up, but the second that you press the "dismiss" button, we immediately go back to sleep again. Only after a dozen of alarm clocks go off and the time for the lecture is approaching that we finally have to leave our bed.

This process repeated again and again, and it wasted a lot of precious time for student. If we get up late, we will have to skip the breakfast to get to the course on time. Since I have a chance to design a project in the 110 Honor Section, I want to design a wake-up system to solve the problem once in for all.

Since my 110 Final Project for the regular section uses multiple sensors, memory devises, and display LED arrays, but not Arduino, I want to gain some knowledge about Arduino in the Honor Project while using sensors.

Design Concepts:

After much revision, I decided to use two sensors and 3 active components to build the wake-up system:

Sensor:

1. Sound sensor to detect the alarm clock going off on your phone.
2. Flex sensor to detect presence of the phone.

Active Components:

1. LED lights represents the light in the room.
2. Step motor to control the roll up and down of the curtain.
3. Servo motor to control the angel of the bed.

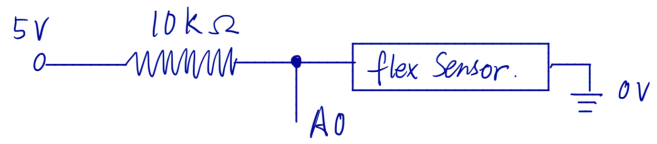
Control Components:

1. Arduino Mega.

Analysis of Components:

Characterization of flex sensor:

In this project, the flex sensor is used as a force detector to detect the presence of the phone. The flex sensor will provide an analog signal to the Arduino. Since the flex sensor has a changing resistance, a voltage divider circuit is used. The analog signal is the voltage across the flex sensor.



A 10k Ω resistor is connected to the flex sensor in series. The Analog input of the Arduino is measuring the voltage across the flex sensor.

(Characterization is shown on the next page)

Sensor status	Resistance Ω	Output Voltage (V)	Sensor power (W)	Resistor power (W)
Straight	14.7k	2.8	0.000533	0.000484
Flexed	46k	3.8	0.000314	0.001444

I have made sure that there is distinctive enough difference between two output voltages since the flex sensor has variable resistance. Also, I have made sure that every component in the circuit will operate within their power ratings under all conditions.

Sound sensor:

Arduino has analog read and digital read, and I have explored the analog read part of the Arduino, so I use digital read for the sound sensor. When the sound level is high enough, the Arduino will mark the value as HIGH, and the value can be used in the program as conditions of codes.

Servo motor:

Servo motor is a type of motor that is commonly used in high precision limited revolting angle environment. The servo motor in the project enables 180-degree angle movement with precision to 1 degree, which is an ideal motor to tilt the bed since I don't need more than 180 degree of movement, but I need high controllability.

Step motor:

Step motor is a type of motor that is commonly used in high precision continuous revolving environment. The step motor used in the project enables unlimited angle revolving in both direction with 1.8-degree precision per step. So, a step motor is ideal for rolling up the curtain – I want to roll in both direction in controllable manner to avoid damaging the curtain.

Unlike the servo motor, a step motor not only need Arduino to control, but also a H-bridge. I use a H-bridge in the project with all for channels enabled, so the stepper can revolve in either direction. The H-bridge also provide the stepper with 12V of power instead of 5V from the Arduino, which also cannot withstand high current.

Control unit:

The entire system is controlled by an Arduino Mega, and various components and sensors are connected to it. Sensors provide inputs to Arduino, and the program written on the Arduino takes the input to provide a response output to system's active components.

Working Flow Chart:

Preparation:

Place a phone with an alarm on the charging dock, the flex sensor should be flexed. Curtain is closed; light is off; you are on the bed sleeping.

Stage one:

The alarm goes off, triggered the sound sensor, which turns on the LED light, and send signal to step motor to roll up the curtain.

Stage two:

You get up from the bed to reach the phone on the charging dock placed on the table to turn off the alarm. The flex sensor will send signal to turn the stepper motor to tilt your bed, so you cannot get back to your bed to sleep again.

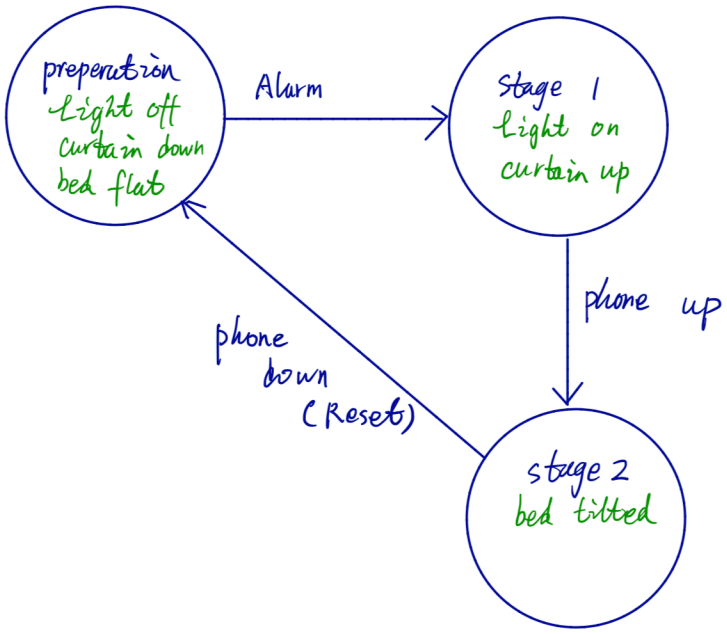
Reset:

Placing the phone back to the dock to charge in the night will trigger the reset mode. The light will be turned off, the curtain will be roll down, the bed will be leveled, and sleep is possible.

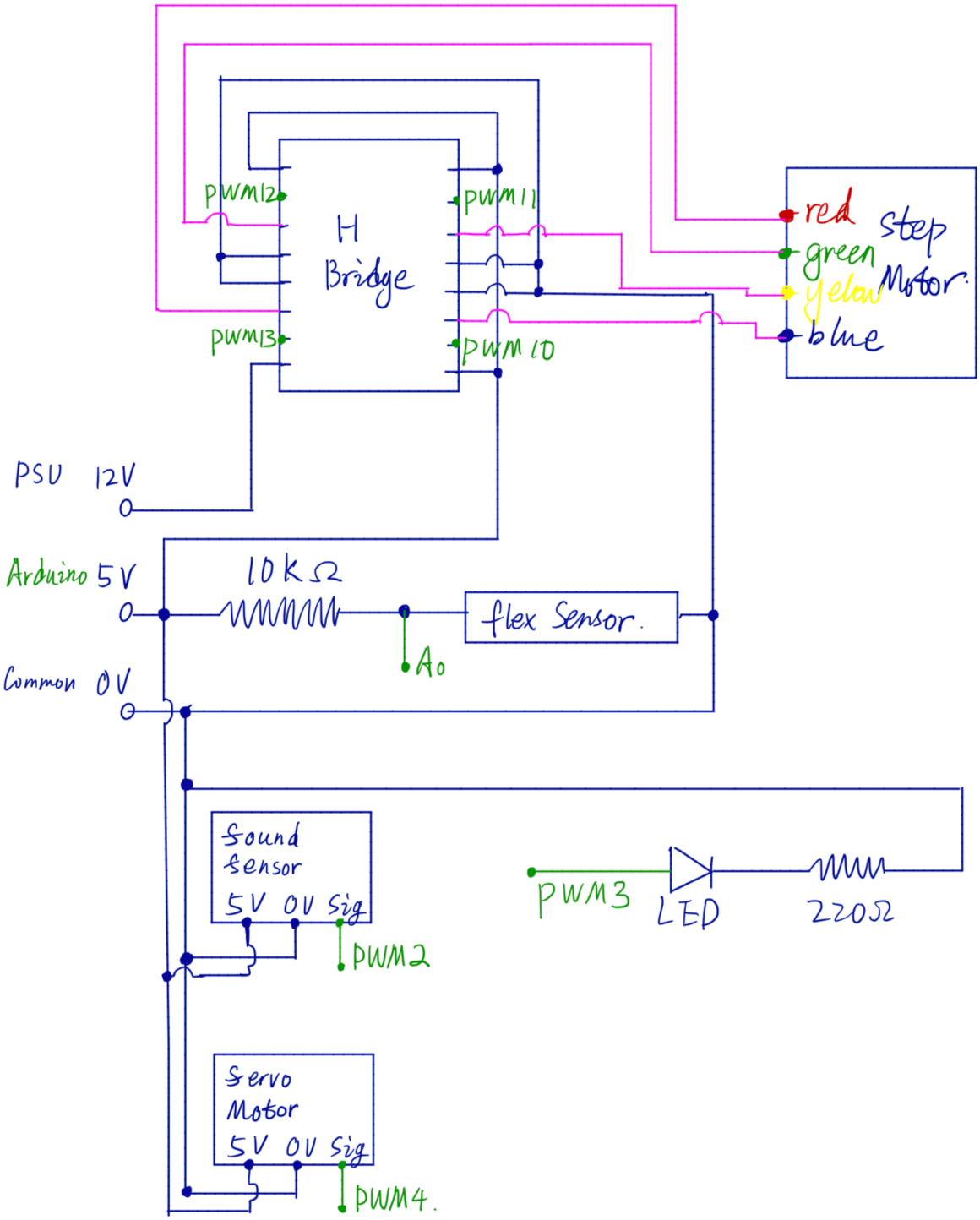
The current stage after reset will be back to preparation state.

Design Description:

Block Diagram:



Circuit Schematics:



Explanation of design considerations and difficulties:

When Dealing with the varying resistance components in the circuits, it is important for us to make sure that under the circuit condition such as the constant voltage across the series connection in this case, no matter how the component change its resistance within the operating range, every component in the circuits will operate within their power rating. Otherwise, we have to change the design to give the circuits higher power ratings.

When writing the program for the Arduino to control the project, I found out that the circuits can enter some unexpected states due to the unexpected inputs. For example, when the curtain is rolling up and you pick up the phone, the bed will not be tilted, and sometimes the curtain will roll down again when you pick up the phone.

Also, when I try to add a reset functionality, everything went crazy, and appears to be random. But after much deliberation and applying knowledge from Finite State Machine, I was able to solve the logical problem, which the code will be explained in detail in the next section.

Program explanation:

<The following program is written by Jiarui Zou.>

```
//Include the library for step motor and servo motor.
#include <Stepper.h>
#include <Servo.h>

// pin assignments
const int soundSensor = 2;
const int LED = 3;
const int servoMotorPin = 4;
const int weightSignal = A0;
const int LEFT_1 = 12;
const int LEFT_2 = 13;
const int RIGHT_1 = 11;
const int RIGHT_2 = 10;

// stepper motor initialization
const int stepsPerRevolution = 360 / 1.8; // change this to fit the number of steps per revolution for your motor
Stepper myStepper (stepsPerRevolution, LEFT_1, LEFT_2, RIGHT_1, RIGHT_2);
int stepCount = 0;

// servo motor initialization
Servo servo;
int servoMotorPos = 0;

void setup() {
  pinMode (soundSensor, INPUT);
  pinMode (LED, OUTPUT);
  myStepper.setSpeed(80);
  servo.attach(servoMotorPin);
  servo.write(servoMotorPos);
```

```
  delay(2000);
}

//Main loop to be executed repetitively
void loop() {
  //In preparation state, constantly reading the input from the sound sensor, jump to the first state if read high
  int statusSensor = 0;
  while (1) {
    statusSensor = digitalRead (soundSensor);//Digital read for the sound sensor
    if (statusSensor == 1) {
      break;
    }
  }
  digitalWrite(LED, HIGH);//Turn on the LED
  myStepper.step(2100);//Roll up the curtain

  //In first state, constantly reading the input from the flex sensor, jump to the second state if phone is removed
  int phoneUp = 0;
  while (1) {
    phoneUp = analogRead (weightSignal);
    if (phoneUp >= 510 && phoneUp <= 650) {
      break;
    }
  }
  while (servoMotorPos < 91) {//tilt the bed to specified angle
    servo.write(servoMotorPos);
    delay(5);
    servoMotorPos++;
  }

  //In second state, constantly reading the input from the flex sensor, jump to the preparation state if phone is
placed back to the charging dock
  while (1) {
    phoneUp = analogRead (weightSignal);
    if (phoneUp < 510 || phoneUp > 650) {
      break;
    }
  }
  digitalWrite(LED, LOW);//Turn off the light
  myStepper.step(-2100);//Roll down the curtain
  while (servoMotorPos > 0) {//Flatten the bed
    servo.write(servoMotorPos);
    delay(5);
    servoMotorPos--;
  }

  //The loop will be executed again, starting from the preparation state
}
```

Part of the malfunctioning program without using the FSM concept:

```
//   while (1) {
//      int statusSensor = digitalRead (soundSensor);
//      if (statusSensor == 1) {
//         digitalWrite(LED, HIGH);
//         myStepper.step(2100);
//         while (1) {
//            int phoneUp = analogRead (weightSignal);
//            if (phoneUp >= 510 && phoneUp <= 650) { //2.5v - 3.2v
//               while (servoMotorPos < 181) {
//                  servo.write(servoMotorPos);
//                  delay(5);
//                  servoMotorPos++;
//               }
//               break;
//            }
//         }
//         break;
//      } else {
//          digitalWrite(LED, LOW);
//      }
//   }
//   while (1) {
//      int phoneUp = analogRead (weightSignal);
//      if (phoneUp < 510 || phoneUp > 650) {
//         digitalWrite(LED, LOW);
//         myStepper.step(-2100);
//         break;
//      }
//   }
```

Lesson Learned:

Rather than write the program and add functionality over time, it is better to just design the functionality in detain as a whole, works out the input and output of different stages, and then implement them. This strategy avoids cluttered codes such as repetitive loops, and the code written is easier to debug and modified if a new feature needs to be added.

The same logic also applies to building the project. We should first list out the outline, stages, components of the project, then we should draw the flow chart of the project. After testing and characterizing each components and sub circuits, we should plan the board accordingly to avoid flying wires and cluttering components.

I used a small board for the project at first, so adding functionality gradually results in a mess of the board. There are flying wires all over the place, and the voltage divider shorts itself, which result in a incorrect input. When I have to transplant the project onto a larger board, but I missed a wire, which cost me another 1.5 hours to debug. The result could be much worse – if one of the sensor or chip is burned, I will have no time to get the new parts, and I will not be able to finish my project.

Self-assessment:

The project works as expected, and I learned a lot in addition to the normal 110 lab section. I characterized more sensors and learned how to use the Arduino for the first time. Also, I realize the importance of planning when doing a project, which should be helpful for my future academic career and life.