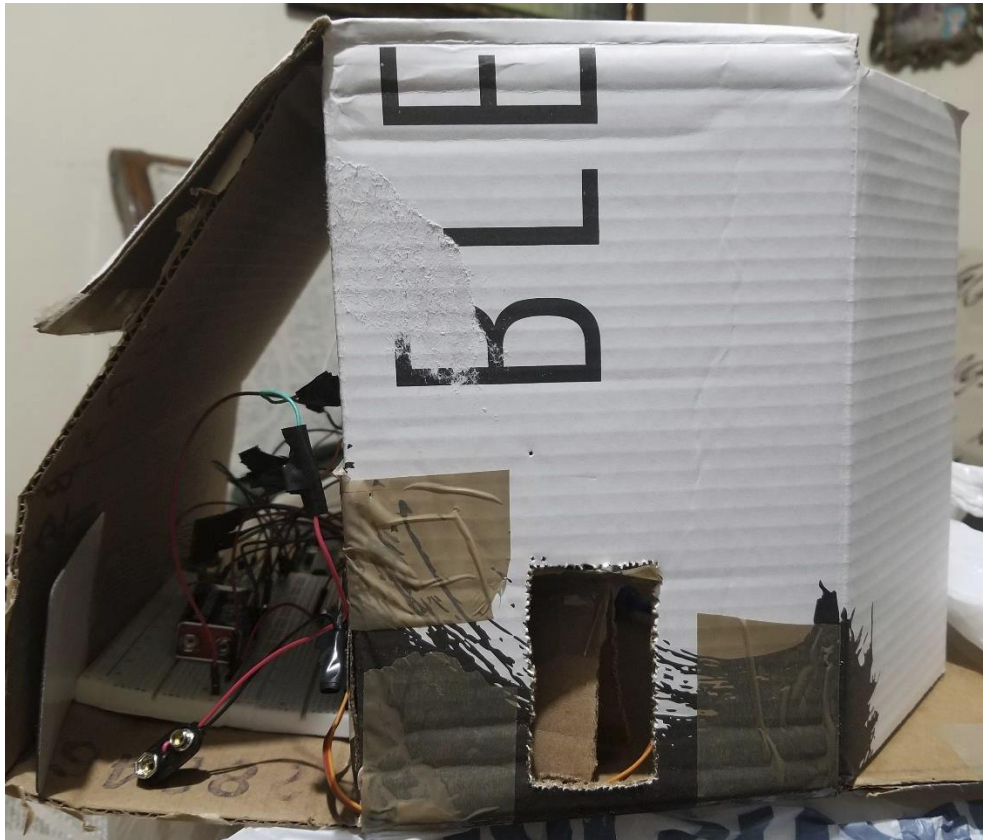




Computer Engineering Technology Department
CET 4811 - Computer Controlled System Design
Year & Semester: Fall 2019
Instructor Name: Professor Farrukh Zia



Capstone Project Report
Automated House System
Date: 12/17/2019

Student Names:
Lordain Bonaventure,
Cristhian Suriel

Table of Contents

<i>Section</i>	<i>Pages</i>
Project Summary	01
Project Management Plan	02
Project Design	02-34
Mechanical design	02-05
Electrical design	05-15
Software Design	15-23
Appendix	24-34
Conclusion	34-35
References	36

1. Project Summary:

The objective of this project is to build an “Automated House System”. We will use Arduino Uno board to implement the software that will make the hardware possible to achieve the goal.

1. A push button switch will be used as input to control a servo motor. In clear words, when you press the switch, the servo motor will be activated in order to open a door.
2. A PIR (Passive Infrared) sensor will be used as an advanced electronic device to measure the infrared light radiating from objects in its field view. The capture of this radiation will control the light in a room. The PIR will also control a speaker connected to the Arduino board. The speaker will emit a type of music to announce an alarm.
3. A 10k Ω - potentiometer will be used as input or voltage divider to control the volume of the speaker. At the same time, the serial monitor of the Arduino board will display the status of the PWM (Pulse Width Modulation).
4. We will use a keypad connected to the Arduino board to enter the information needed to control the servo.

To simulate this project, we will use a prototype of architecture design with a dining-room, kitchen and a bedroom.

Real-world Applications




This type of project of “House Automated Systems” could find many applications in the real-world.

1. Lighting Control: Smart lighting allows you to control wall switches, blinds and lamps. You can schedule the times lights should turn on and off, decide which specific rooms should be illuminated at certain times, select the level of light which should be emitted, and choose how particular lights react through motion sensitivity.
2. Security Systems: With the Internet of Things, it is possible to improve the control of the houses by knowing who is there. Smart locks, like [Kwikset's Kevo](#), a Bluetooth enabled electronic deadbolt, and various connected home security systems, such as [iSmartAlarm](#), offer a variety of features including door and window sensors, motion detectors, video cameras and recording mechanisms. All of which are connected to a mobile device and accessible via the cloud, thus enabling you to access real-time information on the security status of your home.
3. HVAC Regulation
4. Lawn Irrigation Systems
5. LPWA Network: Wide-area, Low-power wireless systems, such as Link Labs' Symphony Links allow end users to employ modules to communicate with a gateway.
6. WI-FI, Bluetooth Smart, Zigbee: You can protect your network by limiting the access to your data from the visitors
7. Welcome the Visitors

2. Project Management:

CET_4811 Computer Controlled Systems II																	
Capstone Project Management																	
Semester: Fall 2019																	
Project Name: Automated House System																	
Group Member Names:																	
Cristhian Suriel																	
Lordain Bonaventure																	
Task / Sub task	Person Responsible		Week														
	Cristhian	Lordain	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1. Mechanical Design	*	*															
2. Electrical Design (Sensor & Motor Interface)		*															
3. Software Design (Control Program)	*																
4. Testing and Debugging	*																
5. Project Demonstration		*															
6. Project Report	*	*															
7. Project Presentation	*	*															

Color Code:

-  Work completed
-  Work remaining
-  Brain Storm or Work in progress

3. Project Design:

I. Mechanical Design:

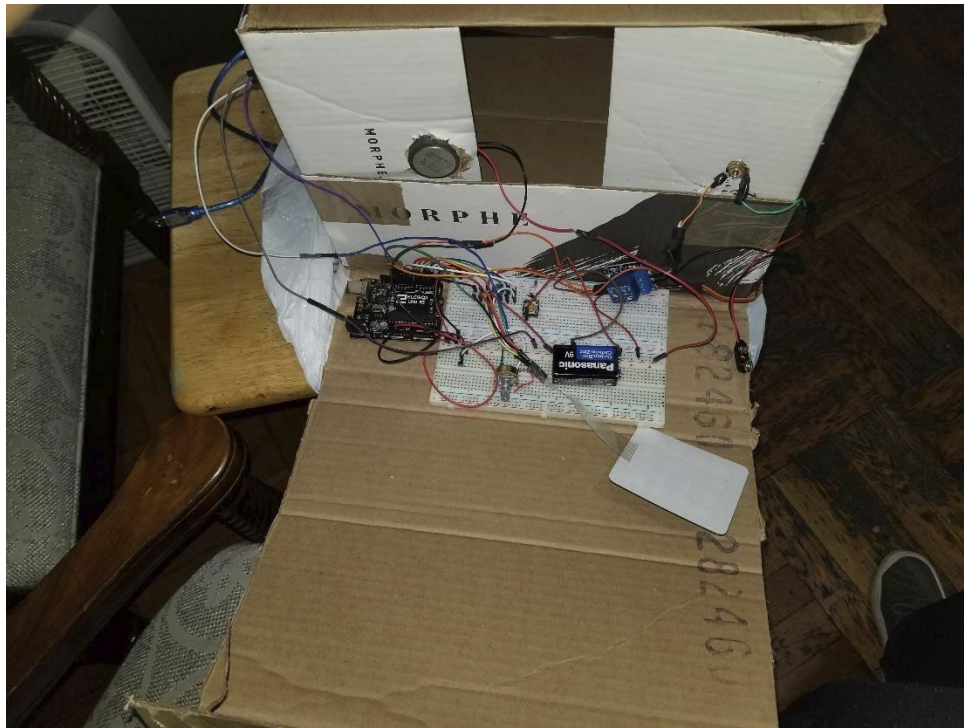
To start making this project all that is needed as far as the mechanical parts are concerned is a square container preferably a box. The following steps will explain the process:

- Close the box and leave one side open. Through this side make holes where all the components will be entering
- Cut away a rectangle on the side of the door and using tape to stick the servo motor to the wall use the same tape to attach the cardboard door to the servo motor

- Insert all the electrical components through the holes and keep them in place by using tape



- Now that the main electrical components are in place organize the cable so that they reach to the side of the make-believe cardboard house
- Wire all the necessary components into the breadboard and connect them to the Arduino



- Finally feel free to use another cardboard in order conceal all the wires by the house effectively making it look nicer



The cardboard used in this project is 10 inches by 10 inches by 1 foot, but feel free to use any size that would work.

One issue that can be encountered during construction is that tape alone might not hold some of the places on the house, this can be fixed by bending the cardboard itself to make easier to attach to itself.

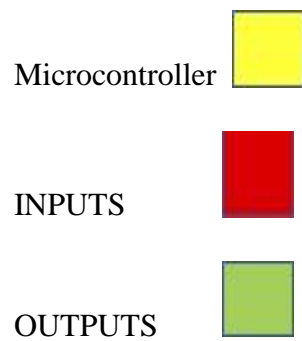
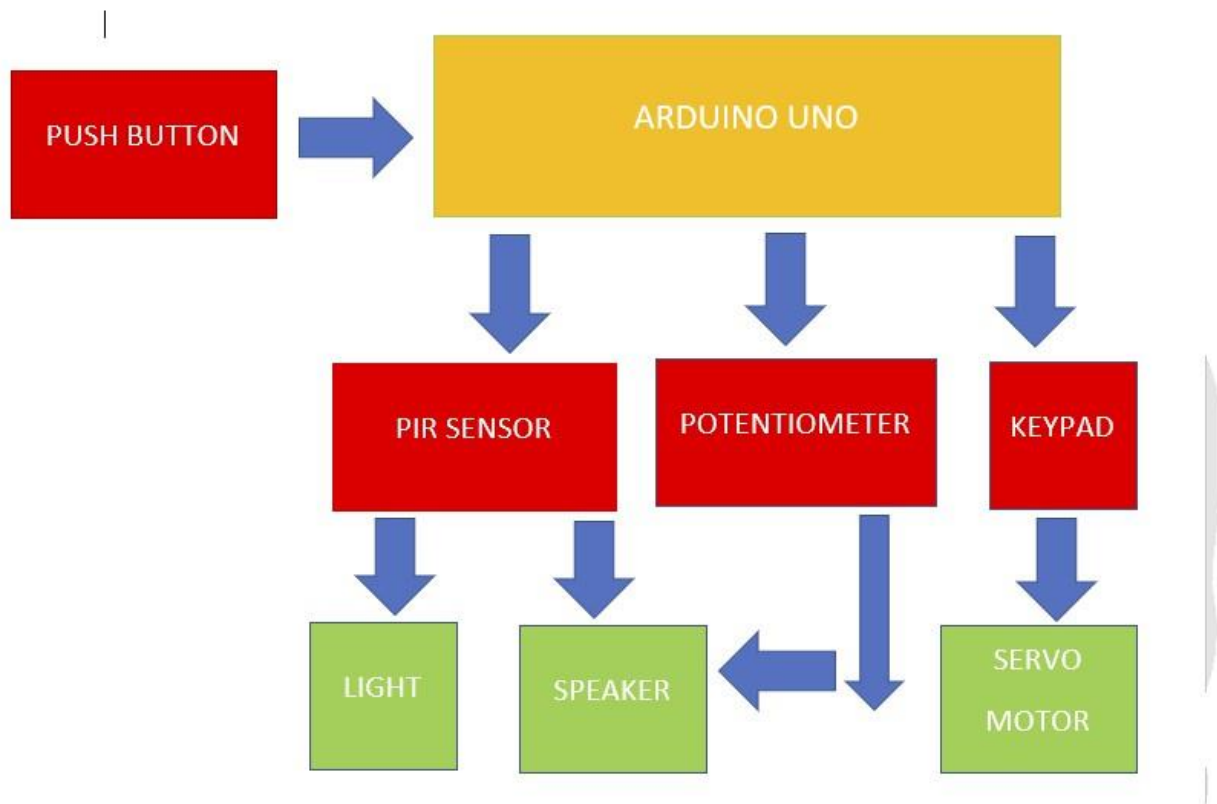
In this lab we ran into the issue that we used the cardboard box to cover the door hole, but failed to notice that the servo attached to it would collide with the walls while opening or would open the wrong way, To avoid this consider testing the servo and marking all the specifics before advancing to the next steps.

II. Electrical Design:

List of Components Used

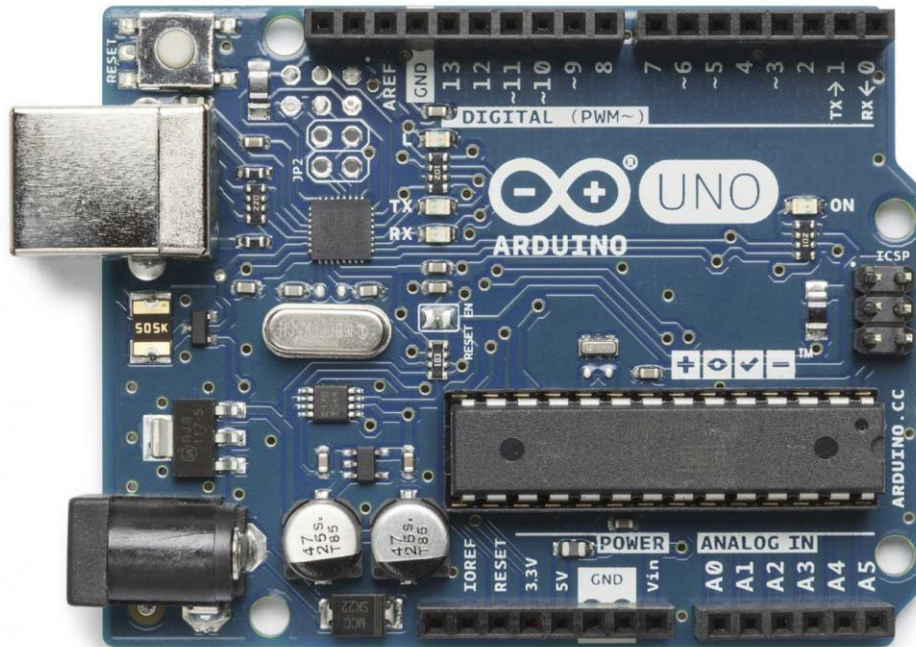
- Arduino UNO Board
- USB Cable
- PIR(Passive Infrared) Sensor
- Push Button Switch
- Keypad
- 10K Ω Potentiometer
- NPN Transistor
- Speaker (5 Ω)
- Servo Motor
- 5V Relay
- 10K Ω Resistor
- Lightbulb

Block Diagram



Embedded Control

The Arduino Uno is used in this project to facilitate the applications of the “House Automated Systems”.

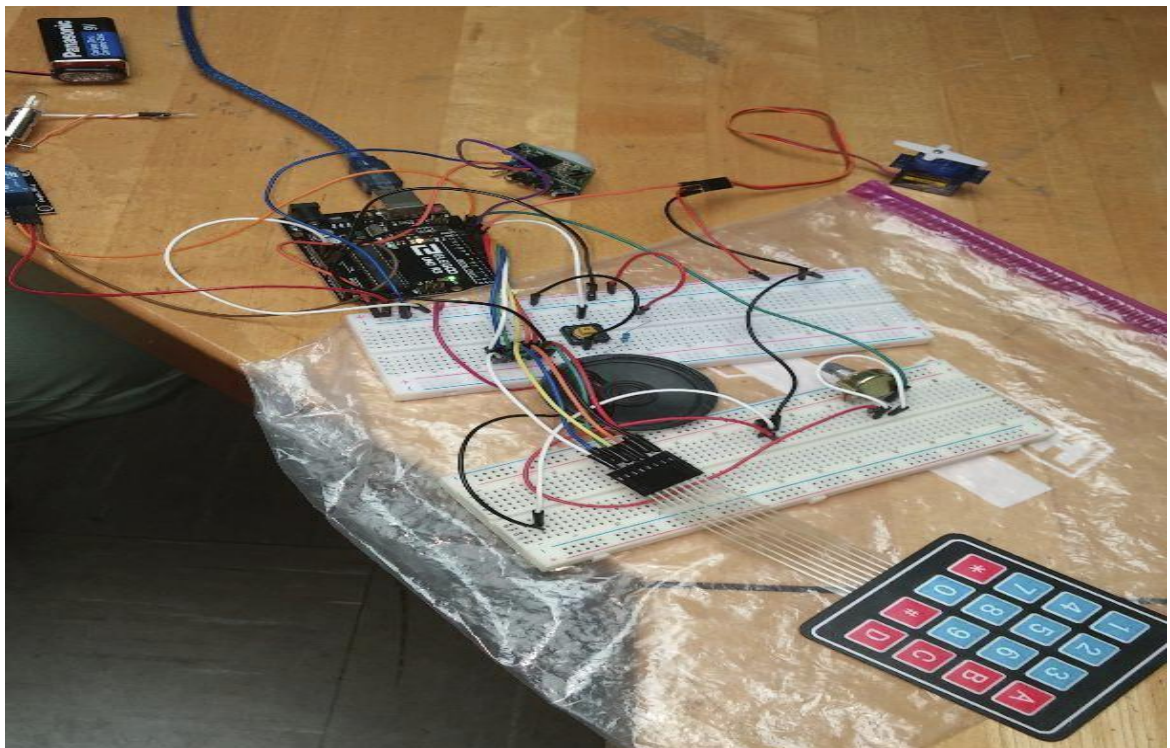
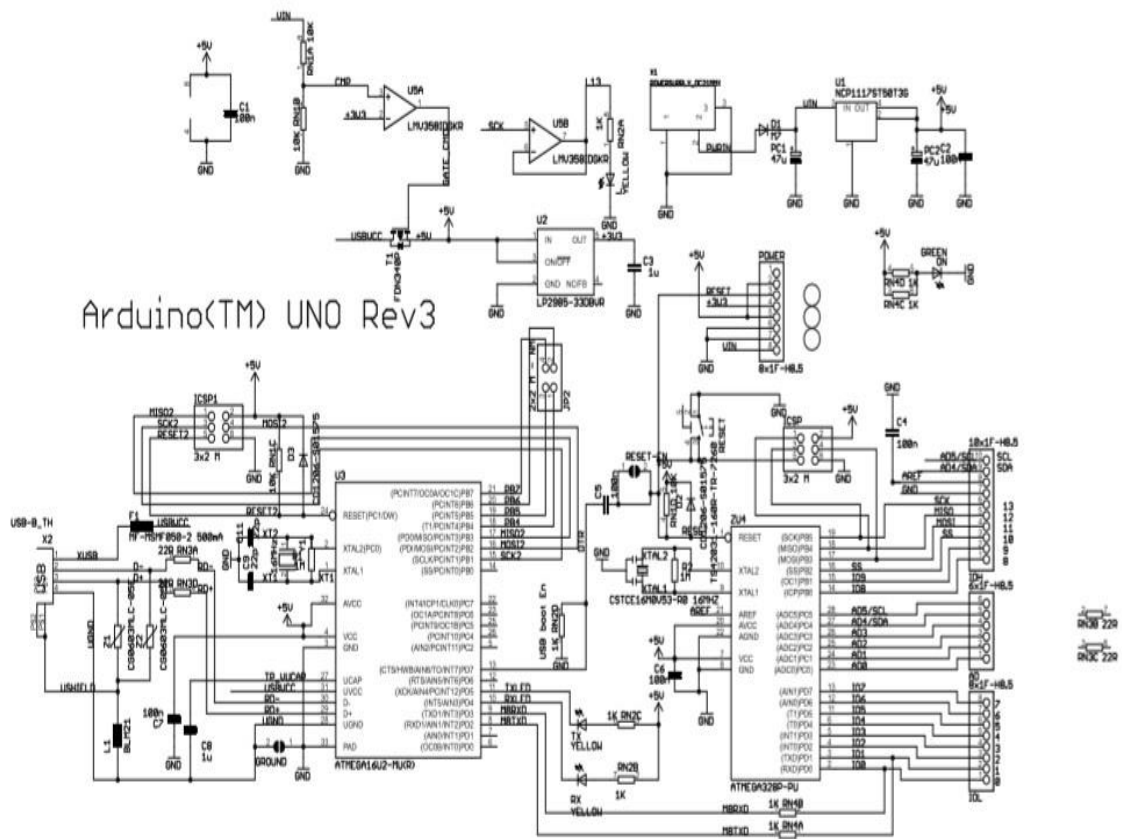


The Arduino Uno is a microcontroller board based on the ATmega328(datasheet). It has 14 digital input/output pins(of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button.

The Arduino Uno can be powered by a computer via a USB cable with a AC-to-DC adapter or battery to get started. Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.

Electrical specifications:

Microcontroller	ATmega328
Operating Voltage	5V Input Voltage (recommended) 7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz



1. Input Sensors

1.1 In this project, we use a push button switch to control the entire “Automated House System”. A push button is an electrical or electromechanical device that is used with Arduino to connect two points in a circuit to turn the system ON and OFF. It can be pressed and released. It is used to interrupt the flow of electrons in a circuit. Switches are essentially binary devices: they are either completely ON(“closed” or completely OFF (“open”). In our case, we use a push button switch as digital input with a pull-down resistor. With a **pull - down resistor** and the button pressed you make a logic state ON, and the button unpressed you make the logic state OFF. The contact current rating of the push button used is 50 mA.

The push-button is connected to pin 10 to the Arduino as interface.

1.2 A 10k Ω potentiometer is used as an analog input to control the volume of a speaker.

A potentiometer is a simple knob that provides a variable resistance, which we can read into the Arduino board as an analog value. The potentiometer will be used as a voltage divider. We connect three wires to the Arduino board. The first goes to ground from one of the outer pins of the potentiometer. The second goes from 5 volts to the other outer pin of the potentiometer. The third goes from analog input pin 12 to the middle pin of the potentiometer.

By turning the shaft of the potentiometer, we change the amount of resistance on either side of the wiper which is connected to the center pin of the potentiometer. This changes the relative "closeness" of that pin to 5 volts and ground, giving us a different analog input. When the shaft is turned all the way in one direction, there are 0 volts going to the pin, and we read 0. When the shaft is turned all the way in the other direction, there are 5 volts going to the pin and we read 1023. In between, analogRead() returns a number between 0 and 1023 that is proportional to the amount of voltage being applied to the pin.

1.3 A PIR (Passive Infrared) sensor is used to control a light and a speaker. The PIR sensor stands for Passive Infrared sensor. This sensor has three output pins: Vcc, Output(Signal) and Ground. This sensor can be used with any platform like Arduino, Raspberry, PIC, ARM etc.

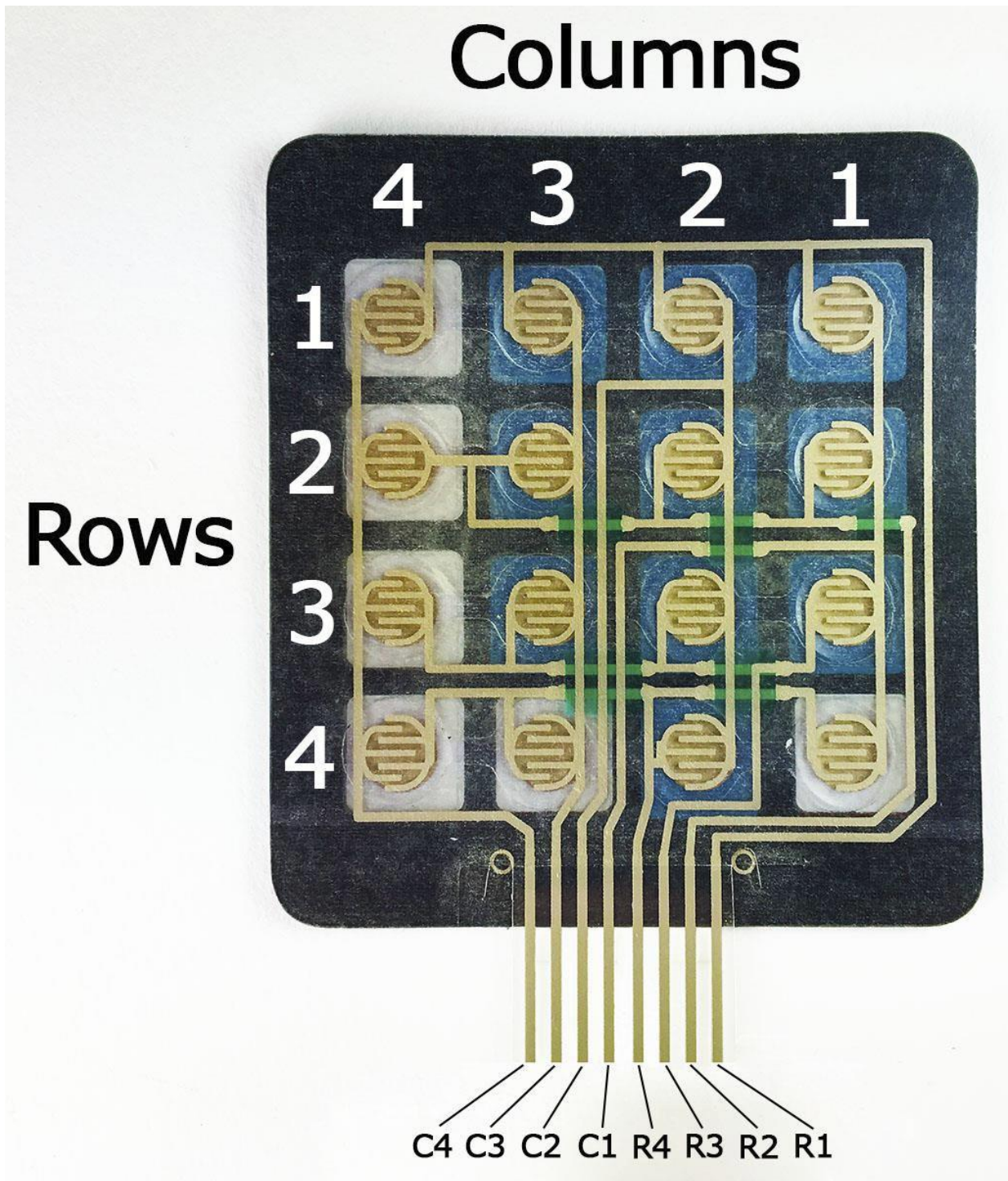
The PIR sensor has the following features:

- Wide range of input voltage varying from 4V to 12V(+5V recommended) ●
- Output voltage is high (3.3VTTL)
- Can distinguish between object movement and human movement
- Has two operating modes-Repeatable(H) and Non-Repeatable(L) but the Repeatable mode is the default mode
- Cover distance of about 120 degrees and 7 meters ●
- Low power consumption of 65mA
- Operating temperature from -20 degrees to 80 degrees Celsius
- Sensitivity can be set using the “Sensitivity control” potentiometer

1.4 A keypad is used to control the servo which can be considered to simulate an entry door.

The buttons on a keypad are arranged in rows and columns. For this project, we use a 4x4 keypad but to economize the pins available on Arduino board, we connect only the three first columns

Beneath each key is a membrane switch. Each switch in a row is connected to the other switches in the row by a conductive trace underneath the pad. Each switch in a column is connected the same way - one side of the switch is connected to all of the other switches in that column by a conductive trace. Each row and column is brought out to a single pin, for a total of 8 pins on a 4X4 keypad:

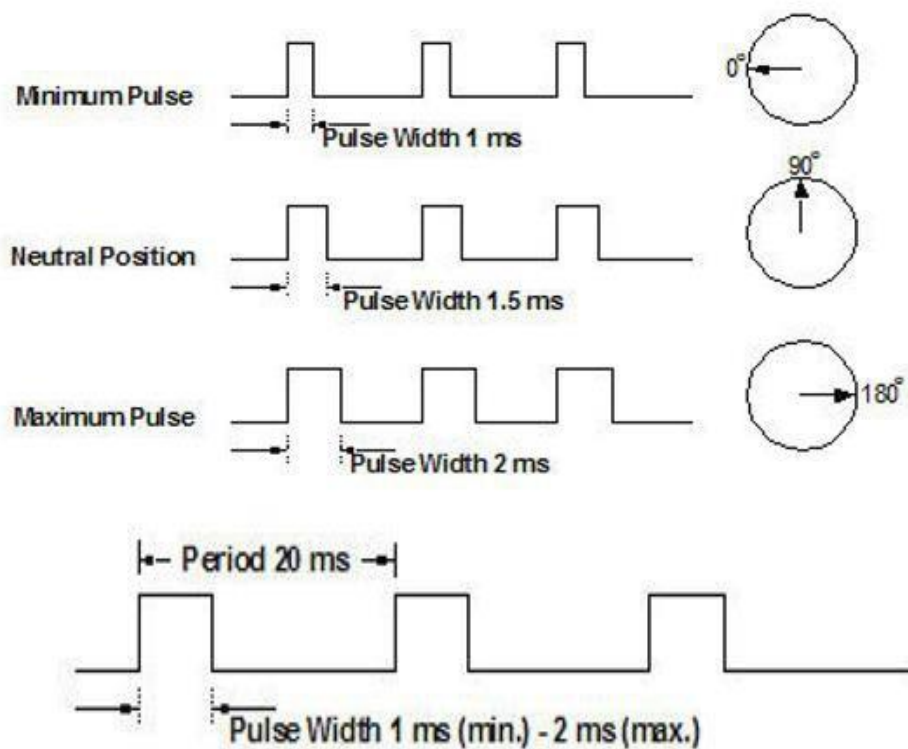


The pins 2, 3, 4, 5, 6, 7, 8 are used as interface to the Arduino.

2. Output Actuators and Devices

3.1. A servo motor is used as an output to simulate an entry door. A Servo is a small device that incorporates a three wire DC motor, a gear train, a potentiometer, an integrated circuit, and an output shaft bearing. Of the three wires that stick out from the motor casing, one is for power, one is for ground, and one is a control input line. The shaft of the servo can be positioned to specific angular positions by sending a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. If the coded signal changes, then the angular position of the shaft changes. Servos work with voltages between 4 and 6 volts. The potentiometer allows the control circuitry to monitor the current angle of the servo motor. Normally a servo is used to control an angular motion of between 0 and 180 degrees. The amount of power applied to the motor is proportional to the distance it needs to travel. A common use of servos is in Radio controlled like cars, airplanes, robots, and puppets.

Servos are controlled by sending them a pulse of variable width. The control wire is used to send this pulse. The parameters for this pulse are that it has a minimum pulse width, a maximum pulse width, and a repetition rate. Given the rotation constraints of the servo, neutral is defined to be the position where the servo has exactly the same amount of potential rotation in the clockwise direction as it does in the counter clockwise direction. It is important to note that different servos will have different constraints on their rotation but they all have a neutral position, and that position is always around 1.5 milliseconds (ms) pulse width.



The angle is determined by the duration (width) of a pulse that is applied to the control wire. This is called Pulse Width Modulation (PWM). The servo expects to see a pulse every 20 ms. The length of the pulse will determine how far the motor turns. For example, a 1.5 ms pulse will make the motor turn to the 90 degree position (neutral position). A 2ms pulse will make the motor turn to the 180 degree position.

3.2. A speaker is used as output in this project to emit a sound whenever a motion is detected by the PIR sensor. A loudspeaker (or loud-speaker or speaker) is an electroacoustic transducer;[1] a device which converts an electrical audio signal into a corresponding sound.[2] The most widely used type of speaker in the 2010s is the dynamic speaker, invented in 1924 by Edward W. Kellogg and Chester W. Rice. The dynamic speaker operates on the same basic principle as a dynamic microphone, but in reverse, to produce sound from an electrical signal. When an alternating current electrical audio signal is applied to its voice coil, a coil of wire suspended in a circular gap between the poles of a permanent magnet, the coil is forced to move rapidly back and forth due to Faraday's law of induction, which causes a diaphragm (usually conically shaped) attached to the coil to move back and forth, pushing on the air to create sound waves. Besides this most common method, there are several alternative technologies that can be used to convert an electrical signal into sound. The sound source (e.g., a sound recording or a microphone) must be amplified or strengthened with an audio power amplifier before the signal is sent to the speaker. Smaller loudspeakers are found in devices such as radios, televisions, portable audio players, computers, and electronic musical instruments.

When the speaker is connected directly to the Arduino via the potentiometer, the sound produced is weak. It does not either with a capacitor. However, when we replace the capacitor by an NPN transistor(amplifier) 2N706, the sound emitted by the speaker increases.

3.3 Light/Relay

A bulb light is used in this project to turn ON whenever a motion is detected by the PIR sensor. We use a lightbulb of 5.6VDC. We also use a 5V relay because the lightbulb cannot be connected directly to the Arduino that provides only 5VDC. The relay has two sides: one receiving the high voltage from AC or from a 9VDC or a 12VDC. This side contains the pins C, NO, NC.

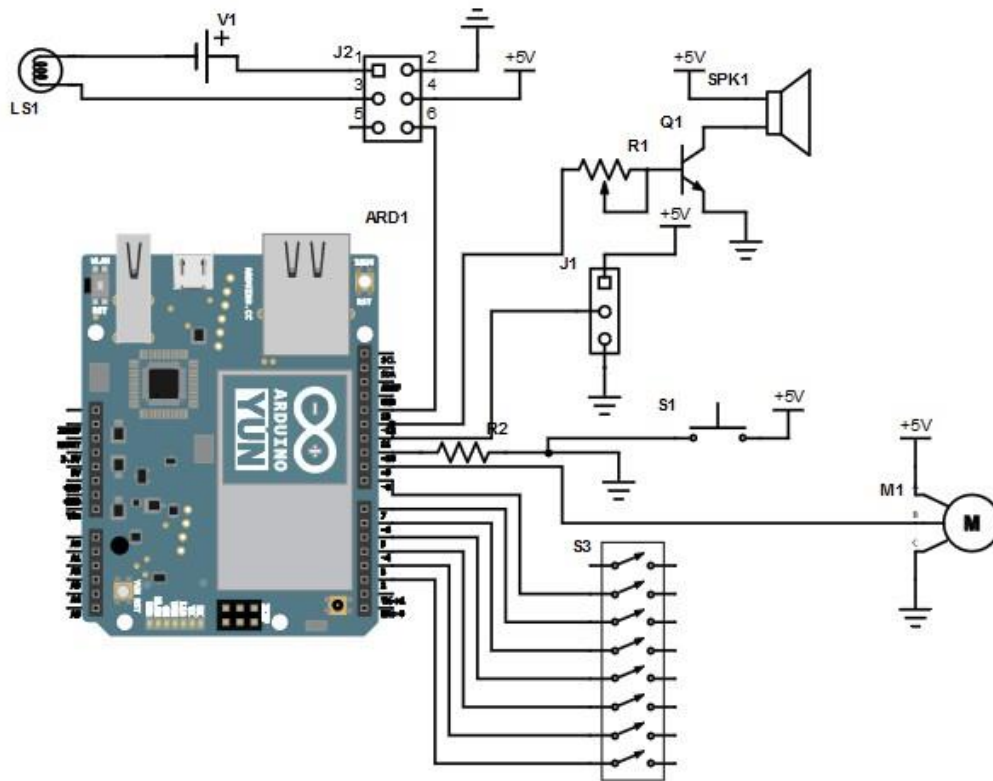
COM = Common Connection-It is the center terminal, and it is hot as power to the load is connected at this terminal.

NC = Normally Closed Connection- It is always in contact with COM, even when relay is not powered. When we trigger the relay, it opens the circuit.

NO = Normally Open Connection- It acts as a switch, since it is open. There will be no contact between COM and NO. When we trigger the relay module, it connects to COM by the electromagnet inside the relay and supply to the load provided, which powers up the light. Thus the circuit is closed until we trigger the state to low in relay.

The other side of the relay that contains the pins Vcc, Signal and Ground are connected to the Arduino. The Vcc pin is connected to 5V on Arduino. The Signal is connected to pin 13 as interface to the Arduino.

3. Input and Output Interfaces / Schematic Diagram



4. Data Communication and Network Interface

Arduino UNO uses three common peripheral interfaces: UART, I2C, and SPI. SPI stands for Serial Peripheral Interface.

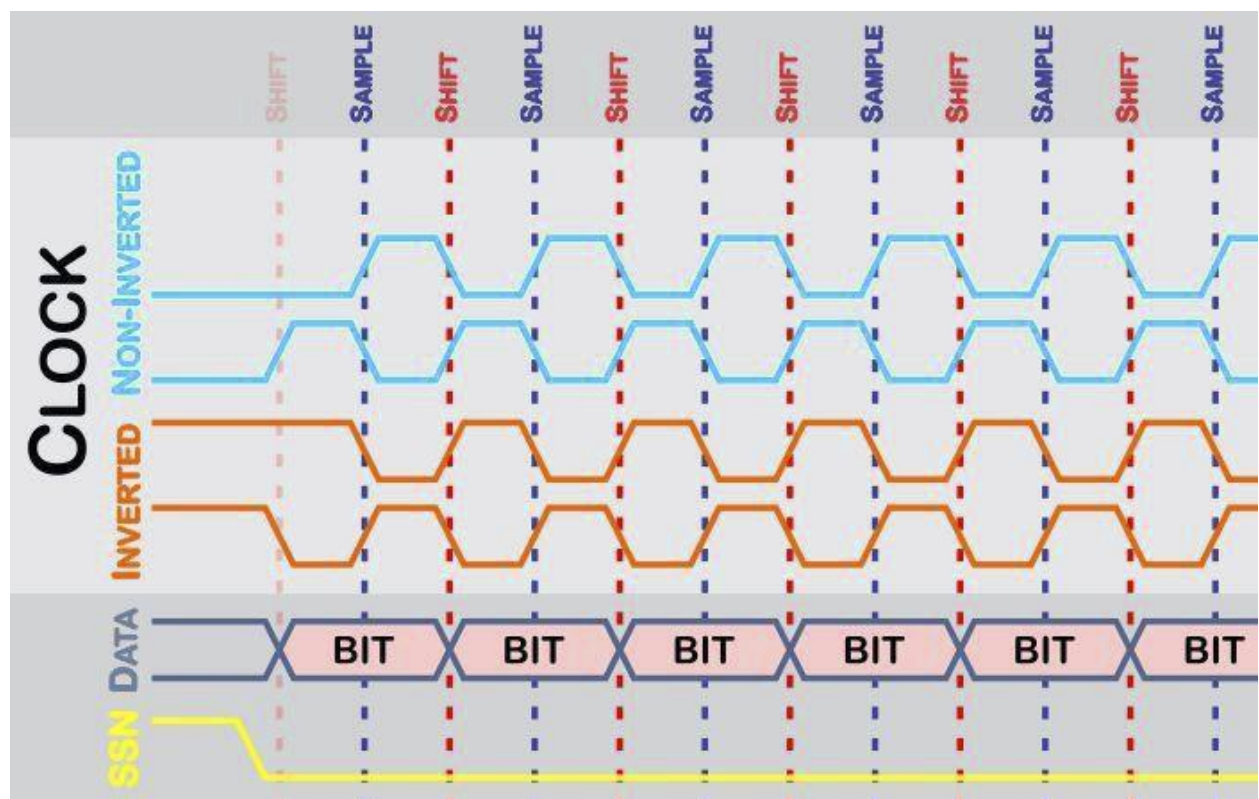
5.1. In this project, we use the SPI, which allows a single master device with a maximum of four slave devices. The following pins are used as SPI: 10, 11, 12, 13. A push button is used as input, and is connected to the SPI pin Slave Select 10 to control the entire Arduino Alarm System. The push button is associated with a pull-down, and every time it is pressed, it turns the entire system ON or OFF depending on the presence or the absence of the owners of the house in order to activate or deactivate the system. The slave-select is usually controlled by the master device. An active slave-select line that the master is sending data from the corresponding slave device.

The SPI is typically much faster than I2C due to the simple protocol, while data/clock lines are shared between devices, each device requires a unique address wire. SPI is commonly used where speed is important such as SD cards and display modules, and when information updates and changes quickly, like with temperature sensors.

5.2. The PIR (Passive Infrared) motion sensor is used as an advanced input sensor to control the relay/light and the speaker. In this case, the PIR used the SPI pin 11 from the Arduino as peripheral interface in order to establish a relationship between the Master Out and Slave In (MOSI). Data leaves the master device and enters the slaves device.

5.3. A 10K potentiometer is used as an analog input to control the volume of a speaker. The Potentiometer is connected to SPI pin 12, where data leaves the slave device and enters the master device (MISO).

5.4. The SPI pin 13 is used where the Arduino controls the 5V relay in order to facilitate the connection of the lightbulb. The serial clock is controlled by the master device. A new data bit is shifted out on the falling edge of the clock pulse.



6. Power System

To power the Arduino board, we use the USB connector to connect to a computer. We also use connected to the PC to program the features. It is also possible to use a DC power jack with Arduino. In this case, DC can only be used to power the board. You can use any adapter that is center positive and 7 to 12VDC output, but 9V is recommended. The Arduino board has a polarity protection diode to avoid destroying the board if you use a negative tip adapter. Arduino has an onboard 5V power supply. Generally, the

onboard power supply allows you to use any wall adapter that gives 7V to 20V, and will regulate that voltage down to a very clean 5V. The maximum output current you can pull is approximately 0.5 Amps.

The Arduino can use a multiple management subsystem to control multiple inputs. The selection mechanism uses a comparator to select the appropriate power for the board.

This comparator is a digital electronic device that compares two inputs, and then drives the output to either 5V or ground. The output of the comparator is connected to an Arduino p MOSFET that acts as a switch. Depending on the output of the comparator, the MOSFET makes the decision to get its 5V from the comparator or the USB power line. The output of the Arduino p MOSFET then connects to the 5V line at the output of the LDO voltage regulator and to the onboard 3.3V regulator.

In this project, we also use an additional 9VDC to power the relay lamp.

III. Software Design:

Main Program

```
#include <Servo.h> //include servo library
#include <Keypad.h> //includes the library for the keypad
#include "pitches.h" //includes pitches header file

#define light 13 // assign the keyword light to pin 13
#define spkPin 12 // assign the keyword spkPin to pin 12
#define PIR 11 // assign the keyword PIR to pin 11
#define buttonPin 10 // assign keyword buttonPin to pin 10
#define servoPin 9 //pin for the servo motor

int alarmState = false; //Variable for the push button status to open door
int valPIR = 0; //Value the PIR senses
const byte ROWS = 4; //4 rows for the keypad
const byte COLS = 3; //3 columns for the keypad
String password = "369#"; //password to open door
String passwordAttempt = ""; //empty string for the password attempt

//notes in the melody
int melody[] = {
    NOTE_C6, 0, NOTE_C6
};
//the duration of each note: 4 = quarter note, 8 = eighth note, etc...
int noteDurations[] = {
    2, 4, 2
};
//Keyboard layout
char keys[ROWS][COLS] = {
    {'1', '2', '3'},
    {'4', '5', '6'},
    {'7', '8', '9'},
    {'*', '0', '#'}
};
```

```

byte rowPins[ROWS] = {8, 7, 6, 5}; //connect to the row pins of the keypad
byte colPins[COLS] = {4, 3, 2}; //connect to the column pins of the keypad

Servo myServo;
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

////////////////////Setup Code////////////////////

void setup() {
  Serial.begin(9600);
  pinMode (buttonPin, INPUT);
  pinMode (PIR, INPUT);
  pinMode (light, OUTPUT);
  myServo.attach(servoPin); }

////////////////////Main Code////////////////////

void loop() {
  doorCommand(); //This function checks a password to open the door
  checkAlarm(); //This function checks if the alarm is set or not
  valPIR = digitalRead(PIR); //reads the value the PIR is receiving
  Serial.println (valPIR); //prints the value to the serial monitor
  Serial.println (alarmState);
  //if the PIR detects someone turn the lights on
  if (valPIR == HIGH) {
    digitalWrite (light, HIGH);
    //if the alarm is set it will sound
    while (alarmState == 1) {
      alarmOn();
      checkAlarm();
    }
  }
  //if the PIR doesn't detect someone turn the lights off
  if (valPIR == LOW)
    digitalWrite (light, LOW);
}

////////////////////Functions////////////////////

void doorCommand() {
  char key = keypad.getKey(); //get the character from keypad
  if (key) {
    //Serial.println(key);
    passwordAttempt += String(key); //add character to string attempt
  }
  if (passwordAttempt.equals(password)) {
    myServo.write(90); //set servo to midpoint (open door)
    delay (3000); //precaution delay
    passwordAttempt = "";
  }
  else {
    myServo.write(0); // servo position is 0 (door closed)
    delay (1); //door open delay
  }
}

```

```

void alarmOn () {
    // iterate over the notes of the melody:
    for (int thisNote = 0; thisNote < 3; thisNote++) {

        // to calculate the note duration, take one second divided by the note
        type.
        //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
        int noteDuration = 1000 / noteDurations[thisNote];
        tone(spkPin, melody[thisNote], noteDuration);

        // to distinguish the notes, set a minimum time between them.
        // the note's duration + 30% seems to work well:
        int pauseBetweenNotes = noteDuration * 1.30;
        delay(pauseBetweenNotes);
        // stop the tone playing:
        noTone(spkPin);
    }
    delay (1000);
}

void checkAlarm () {
    //Checks if the alarm is activated or not
    if (digitalRead(buttonPin) == true) {
        alarmState = !alarmState;
    }
    while (digitalRead(buttonPin) == true);
    delay(50);
}

```

Pitches header

```

/*****
 * Public Constants
 *****/

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98

```

```
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
```

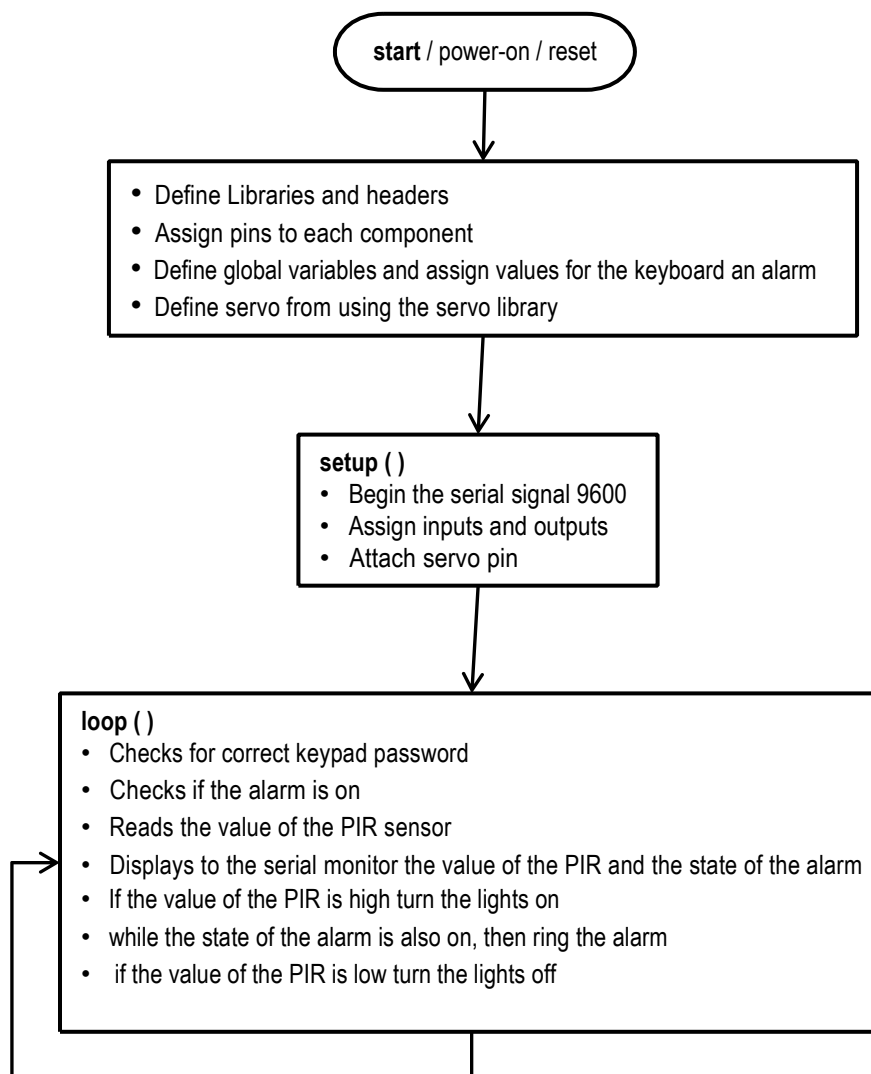


```

#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978

```

Flow-Chart



Analysis

```
#include <Servo.h> //include servo library
#include <Keypad.h> //includes the library for the keypad
#include "pitches.h" //includes pitches header file
```

This code includes all the necessary headers and libraries. Note that the keypad library has to be downloaded separately

```
#define light 13 // assign the keyword light to pin 13
#define spkPin 12 // assign the keyword spkPin to pin 12
#define PIR 11 // assign the keyword PIR to pin 11
#define buttonPin 10 // assign keyword buttonPin to pin 10
#define servoPin 9 //pin for the servo motor
```

This piece of code is dedicated to defining where pins 9-13 will be assigned to

```
int alarmState = false; //Variable for the push button status to open door
int valPIR = 0; //Value the PIR senses
const byte ROWS = 4; //4 rows for the keypad
const byte COLS = 3; //3 columns for the keypad
String password = "369#"; //password to open door
String passwordAttempt = ""; //empty string for the password attempt

//notes in the melody
int melody[] = {
    NOTE_C6, 0, NOTE_C6
};
//the duration of each note: 4 = quarter note, 8 = eighth note, etc...
int noteDurations[] = {
    2, 4, 2
};
//Keyboard layout
char keys[ROWS][COLS] = {
    {'1', '2', '3'},
    {'4', '5', '6'},
    {'7', '8', '9'},
    {'*', '0', '#'}
};
byte rowPins[ROWS] = {8, 7, 6, 5}; //connect to the row pins of the keypad
byte colPins[COLS] = {4, 3, 2}; //connect to the column pins of the keypad

Servo myServo;
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
```

This piece of code declares variables and assigns values to these variables that will later be used in the main code. Note the comment at each important line of code explaining what each does.

```

void setup() {
  Serial.begin(9600);
  pinMode (buttonPin, INPUT);
  pinMode (PIR, INPUT);
  pinMode (light, OUTPUT);
  myServo.attach(servoPin);
}

```

This is the set-up of the main code. It could be said that the previous step could've been inside this part but for the purpose of keeping things neat it was left short. Here the serial function is enabled, the PIR and the light are defined as inputs and outputs, and using the servo library to attach the servo to its respective pin.

```

void loop() {
  doorCommand(); //This function checks a password to open the door
  checkAlarm(); //This function checks if the alarm is set or not
  valPIR = digitalRead(PIR); //reads the value the PIR is receiving
  Serial.println (valPIR); //prints the value to the serial monitor
  Serial.println (alarmState);
  //if the PIR detects someone turn the lights on
  if (valPIR == HIGH) {
    digitalWrite (light, HIGH);
    //if the alarm is set it will sound
    while (alarmState == 1) {
      alarmOn();
      checkAlarm();
    }
  }
  //if the PIR doesn't detect someone turn the lights off
  if (valPIR == LOW)
    digitalWrite (light, LOW);
}

```

This is the main part of the program, that loops and repeats all the steps that are required for the program to work. Functions like doorCommand, checkAlarm, and alarmOn are explained in detail in later parts of this report. Using digitalRead this code reads the value of the PIR sensor and uses the value to decide whether or not the alarm will turn on and if the lights inside the house will turn on as well.

```

void doorCommand() {
  char key = keypad.getKey(); //get the character from keypad
  if (key) {
    //Serial.println(key);
    passwordAttempt += String(key); //add character to string attempt
  }
  if (passwordAttempt.equals(password)) {
    myServo.write(90); //set servo to midpoint (open door)
    delay (3000); //precaution delay
    passwordAttempt = "";
  }
  else {
    myServo.write(0); // servo position is 0 (door closed)
    delay (1); //door open delay
  }
}
}

```

This is the function in charge of deciding in what condition will the door open and what will happen once it is. Using the predetermined password from before this code uses the keyword “keypad.getKey” to compare the password put in and the actual password to match.

```

void alarmOn () {
  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < 3; thisNote++) {

    // to calculate the note duration, take one second divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(spkPin, melody[thisNote], noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(spkPin);
  }
  delay (1000);
}
}

```

This is the function that is in charge of playing the right notes to make the speaker go off in a form of alarm. Using basic knowledge of notes and pitches and trial and error it was able to accurately sound like a real alarm although note that any melody can potentially be put together.

```

void checkAlarm () {
  //Checks if the alarm is activated or not
  if (digitalRead(buttonPin) == true) {
    alarmState = !alarmState;
  }
  while (digitalRead(buttonPin) == true);
  delay(50);
  // alarmState = digitalRead(buttonPin);
  // if (digitalRead(buttonPin) == true) {
  //   alarmState = !alarmState;
  //   delay(1);
  // }
}

```

This function was specifically made to check the state of the alarm. Using the if statement, and the NOT boolean expression effectively it is possible for this code to toggle a variable between high and low as the button is pressed.

The serial monitor would display 00 if the PIR is LOW and the alarm deactivated

01 when the PIR is LOW and the alarm activated

10 when the PIR is HIGH and the alarm deactivated

11 when the PIR is HIGH and the alarm activated

Software Troubleshooting

Bugs:

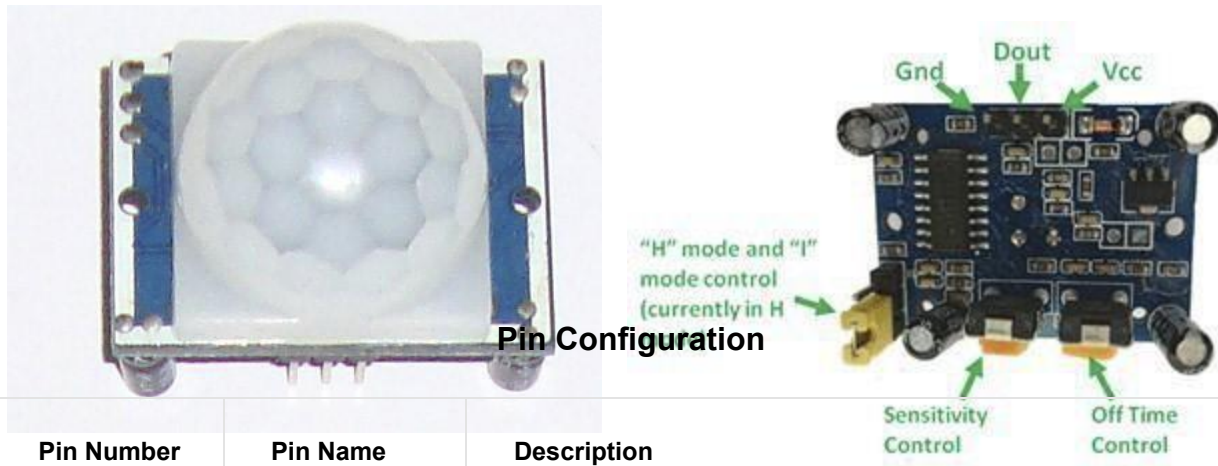
We encountered some bugs such as the PIR sensor always reading HIGH in the serial monitor, the speaker not ringing indefinitely when there was an “intruder” in the house, how to effectively compare to string for the password... etc

Solution Approach:

Due to this project having to do lots of different mechanisms going on at the same time we had to come up with a system to figure out how to determine what wasn't working and why. In order to make this process work we created test codes for each individual component in the circuit, as well as test codes that will test components working together. Using this approach, we would be able to come up with the solution by process of elimination eventually deciding whether the error was in the code or not. For example, for the PIR we decided to use a simple code that will print whether the PIR was detecting something or not, as it turns out it was always detecting something, and the code was so simple that it was easy to determine that the problem lied within the hardware instead of the software.

IV. Appendix:

PIR Sensor

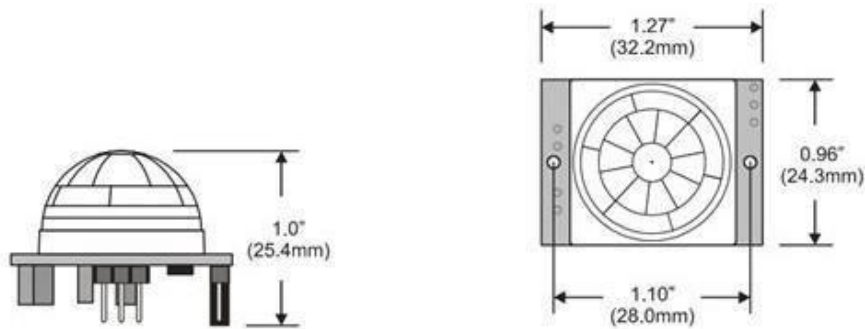


Pin Number	Pin Name	Description
	Vee	Input voltage is +SV for typical applications . Can range from 4.5V- 12V
2	High/Low Ouput (Dout)	Digital pulse high (3.3V) when triggered (motion detected) digital low(0V) when idle(no motion detected)
3	Ground	Connected to ground of circuit

PIR Sensor Applications

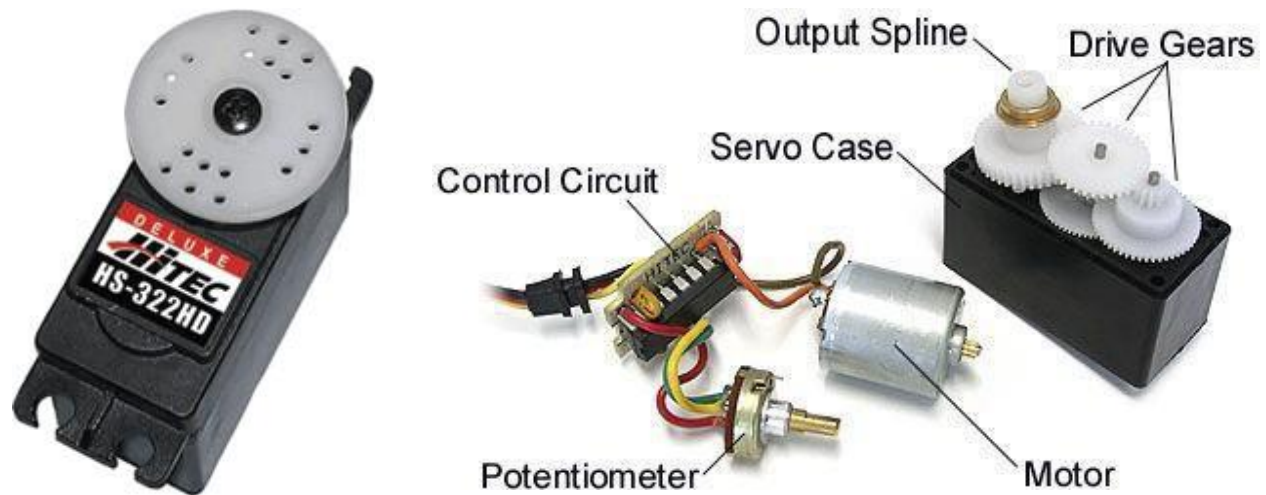
- Automatic Street/Garage/Warehouse or Garden Lights
- Burglar Alarms
- Security cams as motion detectors
- Industrial Automation Control

2D model of the sensor



Servo Motor

A Servo is a small device that incorporates a three wire DC motor, a gear train, a potentiometer, an integrated circuit, and an output shaft bearing. Of the three wires that stick out from the motor casing, one is for power, one is for ground, and one is a control input line. The shaft of the servo can be positioned to specific angular positions by sending a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. If the coded signal changes, then the angular position of the shaft changes. Servos work with voltages between 4 and 6 volts.



The potentiometer allows the control circuitry to monitor the current angle of the servo motor.

The motor, through a series of gears, turns the output shaft and the potentiometer

simultaneously. The potentiometer is fed into the servo control circuit and when the control

circuit detects that the position is correct, it stops the motor. If the control circuit detects that

the angle is not correct, it will turn the motor the correct direction until the angle is correct.

Normally a servo is used to control an angular motion of between 0 and 180 degrees. It is not mechanically capable (unless modified) of turning any farther due to the mechanical stop build on to the main output gear.

The amount of power applied to the motor is proportional to the distance it needs to travel. So, if the shaft needs to turn a large distance, the motor will run at full speed. If it needs to turn only a small amount, the motor will run at a slower speed. This is called proportional control.

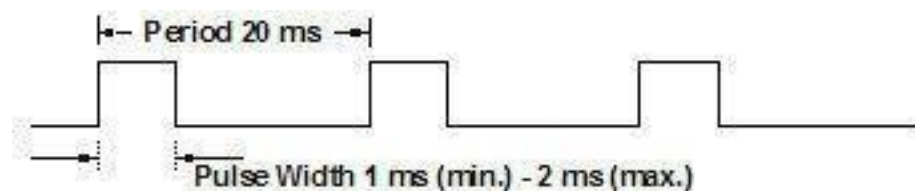
A very common use of servos is in Radio Controlled models like cars, airplanes, robots, and puppets. Servos come in different sizes but use similar control schemes and are extremely

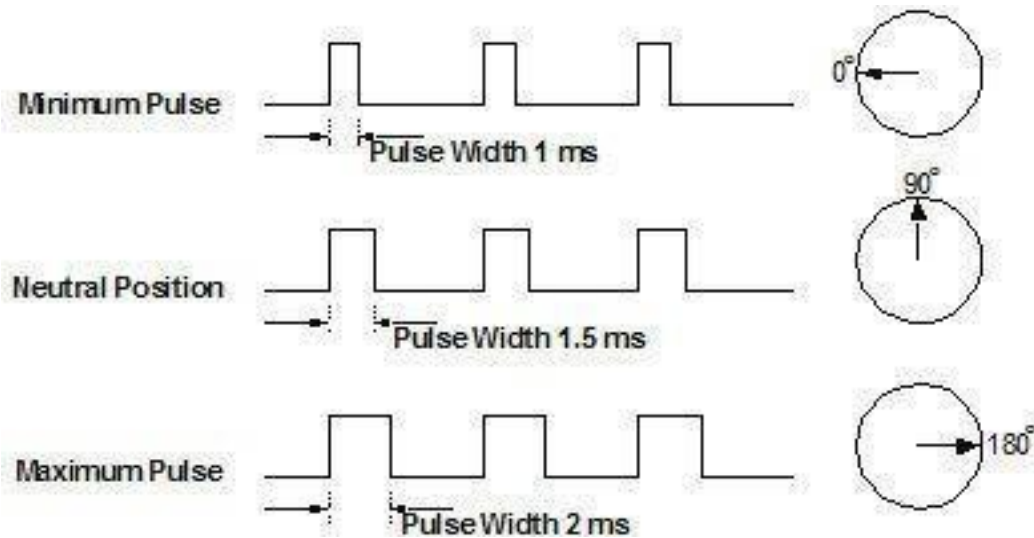
useful in robotics. The motors are small and are extremely powerful for their size. They draw

power proportional to the mechanical load. lightly loaded servo, therefore, doesn't consume much energy.

The gears of a servo vary from servo to servo. Inexpensive servos have plastic gears, and more expensive servos have metal gears which are much more rugged but wear faster. The potentiometer of a servo is the feedback device. The electronics of a servo are pretty much the same in all servos, but the output shaft bearing of a servo has either a plastic on plastic bearing that will not take much side load or a metal on metal bearings that stand up better under extended use, or ball bearings which work best.

Servos are controlled by sending them a pulse of variable width. The control wire is used to send this pulse. The parameters for this pulse are that it has a minimum pulse width, a maximum pulse width, and a repetition rate. Given the rotation constraints of the servo, neutral is defined to be the position where the servo has the same amount of potential rotation in the clockwise direction as it does in the counter clockwise direction. It is important to note that different servos will have different constraints on their rotation, but they all have a neutral position, and that position is always around 1.5 milliseconds (ms) pulse width.





The angle is determined by the duration (width) of a pulse that is applied to the control wire.

This is called Pulse Width Modulation (PWM). The servo expects to see a pulse every 20 ms. The length of the pulse will determine how far the motor turns. For example, a 1.5 ms pulse will make the motor turn to the 90 degree position (neutral position).

When these servos are commanded to move they will move to the position and hold that position. If an external force pushes against the servo while the servo is holding a position, the servo will resist from moving out of that position. The maximum amount of force the servo can exert is the torque rating of the servo. Servos will not hold their position forever though; the position pulse must be repeated to instruct the servo to stay in position.

When a pulse is sent to a servo that is less than 1.5 ms the servo rotates to a position and holds its output shaft some number of degrees counterclockwise from the neutral point. When the pulse is wider than 1.5 ms the opposite occurs. The minimal width and the maximum width of pulse that will command the servo to turn to a valid position are functions of each servo.

Different brands, and even different servos of the same brand, will have different maximum

and minimums. Generally the minimum pulse will be about 1 ms wide and the maximum pulse will be 2 ms wide.

Another parameter that varies from servo to servo is the turn rate. This is the time it takes from the servo to change from one position to another. The worst case turning time is when the servo is holding at the minimum rotation and it is commanded to go to maximum rotation. This can take several seconds on very high torque servos.

Servo Terminology

Coreless Motor - This refers to the armature of the motor. A conventional servo motor has a steel core armature wrapped with wire that spins inside the magnets. In a coreless design, the armature uses a thin wire mesh that forms a cup that spins around the outside of the magnets eliminating the heavy steel core. This design results in smoother operation and faster response time.

Indirect Drive - This refers to the potentiometer inside the servo. The final output shaft (the part that the horn/arm attaches) has to be supported not only near the end but also deep inside the servo case. Indirect drive is when the final output shaft is not dependent on the potentiometer for support inside the gear case. Normally a bushing or bearing supports the load. **Direct Drive** is when the potentiometer plays a supporting role in holding the output shaft in place. Most sub-micro servos are direct drive since they are tight on space and do not have the room for an extra bushing or bearing.

Spline - This is the output shaft of the servo. It is what you attach the servo horns or arms to. Standard Hitec splines are 24 tooth with standard Futaba splines 25 tooth.

Transit Time - This is the amount of time it takes for the servo to move a set amount, usually rated at 60 degrees. Example: A servo with a transit time of .19 sec. to 60 degrees would mean that it takes the servo nearly 1/3rd of a second to rotate 60 degrees.

Torque - This is the maximum power the servo can produce. It is normally rated in inch-ounces. This means that the servo can move this set amount with a 1" arm attached to the output shaft or spline. Example: A servo with a torque rating of 130 in.-oz. can move that amount with a 1 inch arm or slightly over 8 lbs. To convert in.-oz. to pounds of force, divide this rating by 16. Example: $130/16=8.125$ which is in pounds.

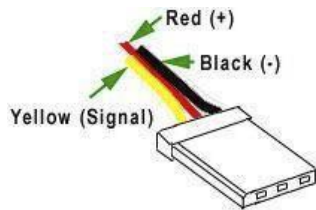
3 or 5 Pole Motors - This refers to the commutator in the motor. The commutator is where the brushes make contact with the armature. The more motor poles the smoother and more accurate the servo will operate. Most servos have either 3 or 5 pole commutators.

Nylon Gears - Nylon gears are most common in servos. They are extremely smooth with little or no wear factors. They are also very lightweight. If your application calls for long duration but not jarring motion, nylon gears are a top choice.

Karbonite Gears - Karbonite gears are relatively new to the market. They offer almost 5 times the strength of nylon gears and also better wear resistance. Cycle times of well over 300,000 have been observed with these gears with virtually no wear. Servos with these gears are more expensive but what you get in durability is more than equaled.

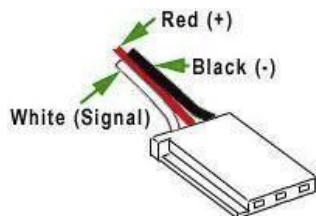
Metal Gears - Metal gears have been around for sometime now. They offer unparalleled strength. With a metal output shaft, side-loads can be much greater. In applications that are jarred around, metal gears really shine. There are two cons to metal gears, weight and wear. First, metal gears are much heavier than both nylon and karbonite gears. Second, metal gears

wear several times that of nylon gears. How quickly depends on the loads that you place on the servo. They will eventually develop a slight play or slop in the gear-train that will be transferred to the spline. It will not be much but accuracy will be lost at some point.

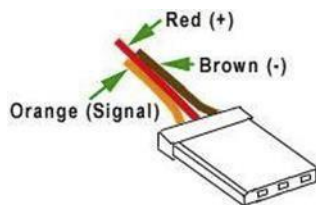


Hitec servos come with the “S” or universal connector. This connector works with any brand of receiver, servo controller or servo extension

Note: This plug can be accidentally plugged into a receiver in reverse. If this happens, no damage will occur, the servo will simply will not work. If you receive a servo and it does not work, make sure that you have plugged it in correctly.



Futaba servos come with the “J” or Futaba connector. This connector has a guide on the side of the plastic plug. This connector will not work with Hitec, JR or Airtronic Z or T servo extensions .



JR plugs are the exact same as the Hitec “S” connectors, except the difference in color coding of the wires

Servo Motor Current Drain Specifications

SERVO TEST SHEET TORQUE OZ/IN = ARM X WT

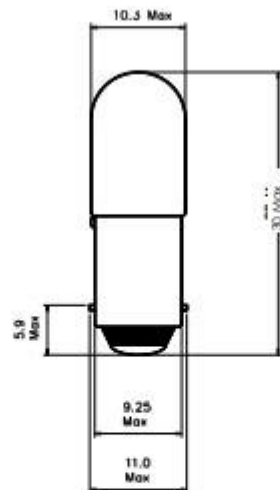
MANUFACTURER AND RATING		MA. CURRENT DRAIN AT % LOAD							MA.
SERVO		TORQUE OZ/IN	IDLE MA	25 % LOAD	50 % LOAD	75 % LOAD	100 % LOAD	STALLED	AVERAGE LOAD
ACE 14G26A		45.0	7.5	25	54	94	215	665	97
ACE 14G26AB		45.0	6.1	27	75	130	310	640	135
AIRT. 94102		50.0	4.5	20	190	310	410	520	232
FUTABA 94510		110.0	8.5	55	290	385	500	720	307
FUTABA 94732		67.0	6.6	65	235	380	415	580	274
FUTABA S114		167.0	9.3	75	330	530	740	910	419
FUTABA S128		48.7	9.5	35	125	245	340	590	186
FUTABA S134		112.6	9.7	145	365	565	780	920	464
FUTABA S134G		173.8	20.0	85	390	505	630	910	402
FUTABA S148		42.0	9.0	45	170	245	325	600	196
FUTABA S9101		41.7	9.2	50	165	245	335	500	199
FUTABA S-34		112.6	9.6	135	360	570	765	910	457
WORLD S-16		180.0	7.5	95	340	520	620	860	394
WORLD S-29		34.0	6.2	20	40	75	170	505	76

Light bulb

Credits to <https://www.alliedelec.com/m/d/cc2e8ce3f0ffb033febe3f9c9b252178.pdf>

Type	Size	Rating	Description
Incandescent	T10	6.3V 150mA	T-3¼ Miniature Bayonet lamp

Dimensions(mm):



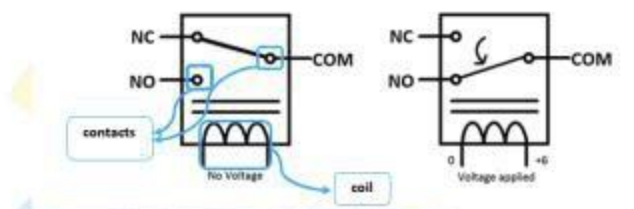
TEST VOLTS	CURRENT (mA)			LIGHT OUTPUT (Lm)			LIFE Hours
	Min.	Nom.	Max.	Min.	Nom.	Max.	
6.3	135	150	165	3.6	4.1	4.7	20000

Notes:

- Lamp base BA9s/13 in accordance with IEC61-1, (BS EN 60061-1) sheet 7004-14-x.
- Lamp marking **CML755 6.3V 150mA**
- Filament shape C-2R
- Bead colour: White or Clear
- Filament in line with pins on base
- Light centre length 20mm nominal
- RoHS and REACH SVHC compliant

Relay module

Credits to https://www.fecegypt.com/uploads/dataSheet/1522335719_relay%20module.pdf

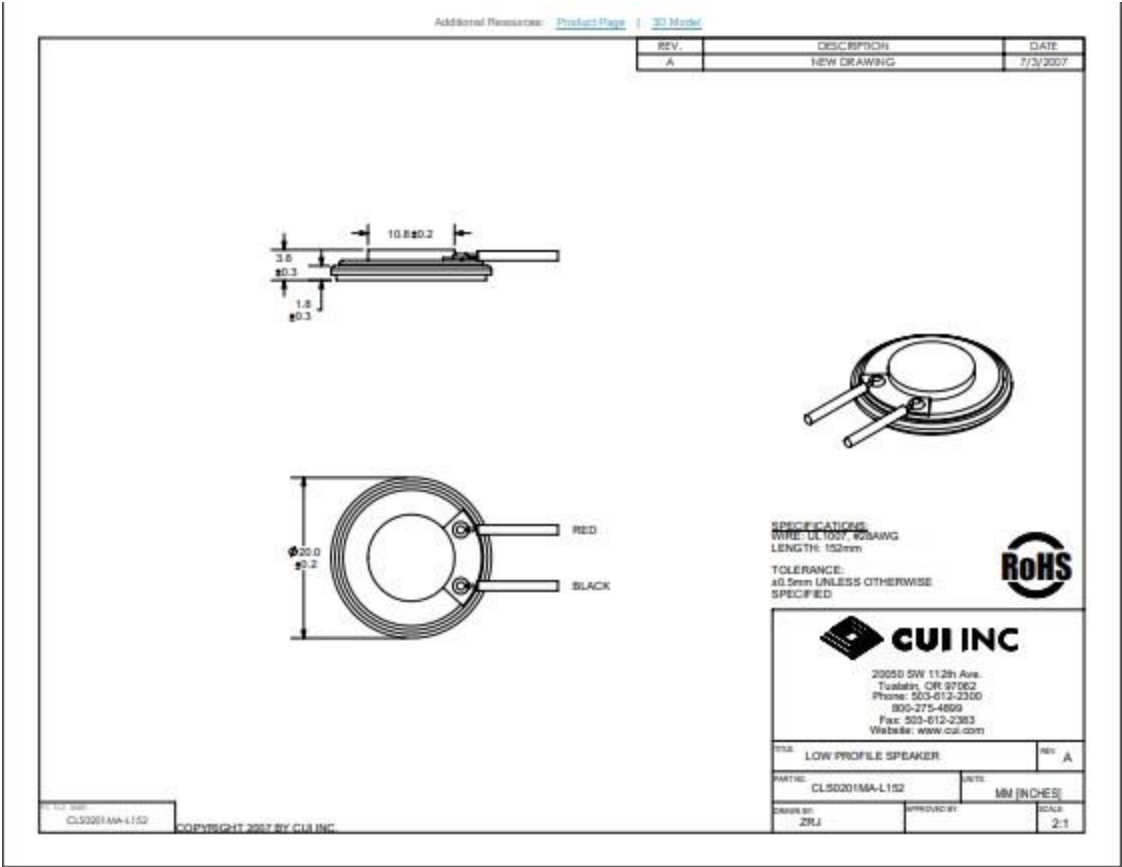


Relay modules 1-channel features

- Contact current 10A and 250V AC or 30V DC.
- Each channel has indication LED.
- Coil voltage 12V per channel.
- Kit operating voltage 5-12 V
- Input signal 3-5 V for each channel.
- Three pins for normally open and closed for each channel.

Speaker

Credits to <https://www.cuidevices.com/product/resource/cls0201ma-l152.pdf>

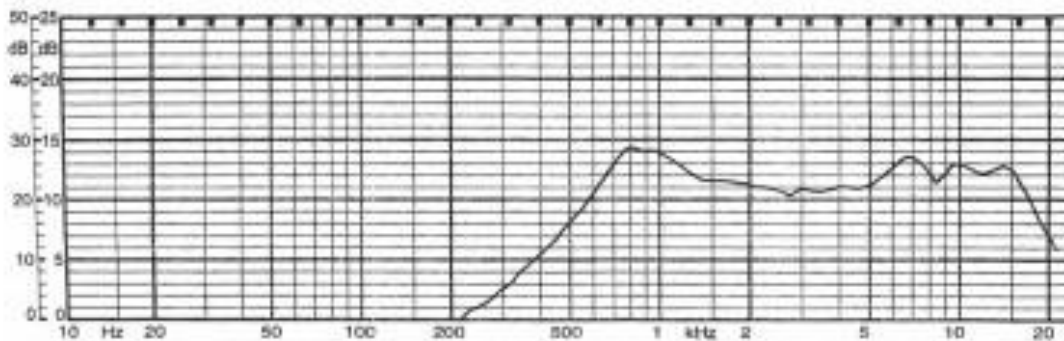


SPECIFICATIONS

parameter	conditions/description	min	nom	max	units
nominal size	20 mm				
impedance	at 1 kHz, 1 V	6.8	8	9.2	Ω
resonant frequency	at 1 V	600	750	900	Hz
sound pressure level	1 W, 50 cm ave., at 0.8, 1, 1.2, 1.5 kHz	81	86	89	dB
response	10 dB max.	Fo		20,000	Hz
input power			0.5	1	W
operation	must be normal at program source		0.5		W
buzz, rattle, etc.	must be normal at sine wave		2		V dc
magnet	size: 8 x 1 mm				
load test	24 hours of white noise at		0.5		W
heat test	20 – 50% RH for 24 hours	58	60	62	$^{\circ}\text{C}$
humidity test	90 – 95% RH for 24 hours	38	40	42	$^{\circ}\text{C}$
RoHS	yes				

FREQUENCY RESPONSE CURVE

parameter	conditions/description
potentiometer range	50 dB
rectifier	RMS
lower limit frequency	20 Hz
scr. speed	100 mm/sec
zero level	60 dB



4. Conclusion:

To complete this project of “House Automated System”, we first planned, and we did some research, then we collected all the components necessary. Secondly, we have built step by step the different parts of the circuit:

- a) We tested the push button switch used as input to control the entire system to turn it ON or OFF. We encountered some difficulty because we used a pull-up resistor connected to the switch, and the system did not work the way we wanted. We changed the configuration to a pull-down resistor, and it worked perfectly. The system remained OFF all the time until we press the push button.

- b) We tested the PIR (Passive Infrared) sensor used as an advanced sensor to control the speaker and the light. The test is done separately. A 10K potentiometer was used with the Arduino to control the volume of the speaker. For the speaker, we tried first a capacitor in series with the speaker, but the sound emitted was very low. Then, we replaced the capacitor by NPN transistor amplifier, and the speaker produced a desirable audible sound every time a motion was detected by the PIR sensor. We tested the PIR sensor to control the light. A 5V relay was used because Arduino can supply only 5V. The lightbulb used is 5.6VDC. An additional 9VDC was used with the circuit.
- c) We tested the keypad (4x4 matrix) used as input to control the servo which is considered as a simulation of a door. For this case, we used the three first columns only, in order to save a pin on Arduino.
- d) After all the separate tests with appropriate codes were performed successfully, we built the entire circuit as a whole as shown in the “Mechanical Design Section”.
- e) We combined all the previous codes, and we uploaded the hardware. We obtained a satisfactory result.

We think that project can be applied in real life, and it can offer substantial advantages to society. This project is not expensive compared to the services provided by the alarm companies. In terms of management, we used an Arduino UNO KIT that cost approximately \$29.00 US dollars, a small speaker for \$5.00, a 5V relay for \$5.50, and a PIR sensor for \$1.00.

Finally we have all the expected results, and we can assume that we reach our objectives by realizing this “House Automated System”.

5. References:

- “Potentiometer.” *Arduino*, 2019, www.arduino.cc/en/tutorial/potentiometer.
- “HC-SR501 PIR Sensor Working, Pinout & Datasheet.” *Working, Pinout & Datasheet*, 18 Sept. 2017, components101.com/hc-sr501-pir-sensor.
- Jos, Ostin. “Controlling AC Light Using Arduino With Relay Module.” *Instructables*, Instructables, 16 Oct. 2017, www.instructables.com/id/Controlling-AC-light-using-Arduino-with-relay-modu/.
- Professor Ziah, F. “New York City College of Technology, Computer Engineering Department.” 2019 Fall CET_4811_E388 Capstone Design Project, Lab_5 Servo Motor Control.
- Pattabiraman, Krishna. “How to Set Up a Keypad on an Arduino.” *Circuit Basics*, 13 Aug. 2018, www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/.
- February, Alex D., et al. “One-Transistor Audio Amplifier for Arduino Projects.” Bryan Duxbury's Blog, 6 Dec. 2013, bryanduxbury.com/2012/01/20/one-transistor.
- “Loudspeaker.” Wikipedia, Wikimedia Foundation, 7 Dec. 2019, en.wikipedia.org/wiki/Loudspeaker.
- Hughes, Mark. “Back to Basics: SPI (Serial Peripheral Interface) - Technical Articles.” *All About Circuits*, 13 Feb. 2017, www.allaboutcircuits.com/technical-articles/spi-serial-peripheral-interface/.
- Hussaini, Umair. “Arduino Uno Power Supply Schematic - Arduino Hardware Core Subsystems.” *Embedded Systems*, 19 Nov. 2019, www.technobyte.org/arduino-uno-power-supply.
- Ada, Lady. “Ladyada's Learn Arduino - Lesson #0.” *Adafruit Learning System*, 14 July 2016, learn.adafruit.com/ladyadas-learn-arduino-lesson-number-0/power-jack-and-supply.