ECE 110/120 Honors Project

Portable Electronic Door Security System v1.0

Team: Jerome Ni/ECE110, Kevin Kim/ECE120, Devarsh Ruparelia/ECE120

Introduction

What have we built? (Statement of Purpose)

We have built an MVP of a Portable Electronic Door Security System. The current stage of the device can accomplish a lot of objectives we started with. It can securely (conceptually) lock and unlock the door. The device can automatically lock the door when the door is closed through IR sensing module and open the door both from inside and outside.

How does the device work? (30000 ft view)

The device detects the right RFID tag (in our case M1 tag (13.5 kHz)) and rotates the servo motor on the back-side platform by 90 degrees. This is the unlock state, wherein a person can get into the room. Once the door is closed, our device detects it and servo motor returns to zero degrees. At this point, the door is in locked state. The person inside has to click a button and interrupt the IR module to open the lock when one intends to go out of the room. This feature is accomplished through use of two NAND(combined to an AND) logic gate.

Benefits of using our device

In most of the Illinois Residence Halls, door locks are not very secure, which has led to robberies in the past at PAR and FAR residence halls. Almost always the robbery resulted because of the fact that residents forgot to lock the door. Our door

Design

Components Used

- Wires
- Arduino Uno
- Servo Motor
- Sparkfun RFID Eval Module (13.5 kHz)
- Xbee Shield Module
- Breadboard
- RFID M1 tags (13.5kHz)
- Button
- IR Emitter
- IR Detector
- Resistors (10k, 4.7k 220 ohms)

System Overview:

Our system has two main components:

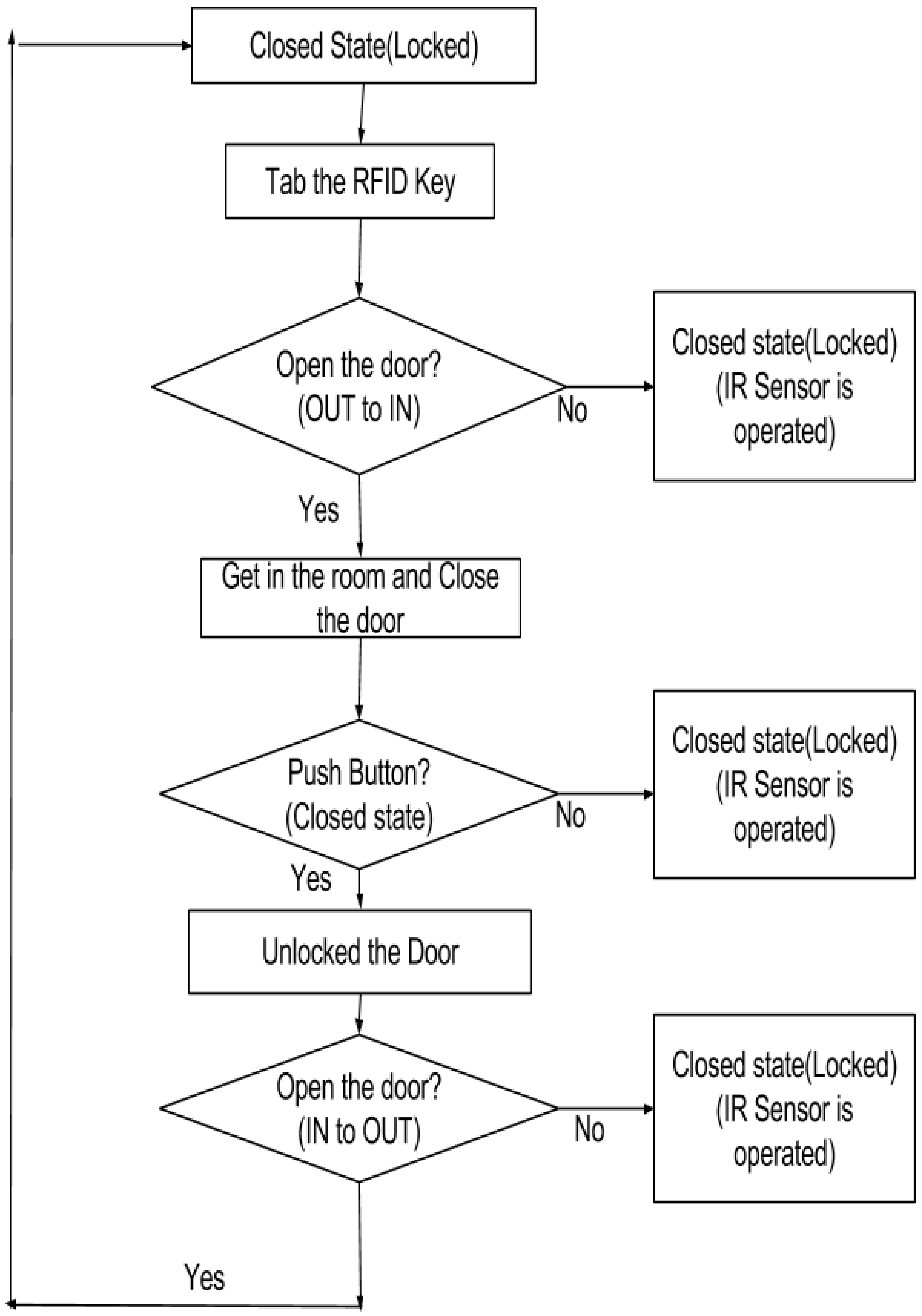
1. Front-side platform (RFID verification)
2. Rear-side platform (Locking system)

Front-side platform verifies if the RFID tag is sensed or not. Then it signals the Rear-side platform to take further action if RFID tag is actually sensed by the detection module in the front. The lock in the rear side changes the state (0 degree - lock state) to (90 degrees - unlock state). Once the person is inside the room, the unlock state is reverted back to lock state. If a person intends to get out of the room, he has to press a button (this button works in conjunction with the IR sensing module). After pressing the button, the person has to get out of the room within 10 seconds (we intend to remove this time barrier in a future version of the device). The lock changes from unlock state to lock state and the room is locked again.

To achieve these tasks we use:

1. Arduino code - to instruct the Rear-side platform to set a new lock system condition if RFID tag is detected.
2. IR sensing module - with button makes the automated door locking possible in our device.
3. Button - to let the person get out of the room conveniently.
4. NAND gate - ensures that Rear-side platform gets control if the person is inside (in order to allow him to get out)

Block Diagram:


Closed State(Locked)
Tab the RFID Key
Open the door? (OUT to IN)
No
Closed state(Locked) (IR Sensor is operated)
Yes
Get in the room and Close the door
Push Button? (Closed state)
No
Closed state(Locked) (IR Sensor is operated)
Yes
Unlocked the Door
Open the door? (IN to OUT)
No
Closed state(Locked) (IR Sensor is operated)
Yes

Functioning of essential components:

We are using a phototransistor and diode system work together as an IR sensing module to both automatically close the door and open the door when it is closed. The first analog pin of the Arduino is read between the phototransistor and the resistor connected in series, ,which gives a high voltage as it sees infrared light from the emitter and a low voltage if it doesn't, so that if the diode on the wall matches the phototransistor on the door, the motor will turn to closed position.

We are using an a NAND gate chip and a button to force open the lock. The first input is read between the phototransistor and the resistor connected in series, ,which gives a HIGH as it sees infrared light from the emitter and a LOW if it doesn't. The second input is read from the button, which connects to GND if the button is not pressed(LOW) and +5V if the button is pressed(HIGH).
The output of this NAND gate also acts as the two inputs of another NAND gate on the chip to form an AND gate. The output of the second NAND gate is read by Arduino analog pin, which forces open the lock when it sees a high voltage.We allow five seconds for the user to push open the door. If the user doesn't do anything, the door will close again.
Combined input and output as follows:

	IR detector sees IR light 1	IR detector Doesn't see IR light 0
Button Pressed 1	1 Motor to open position	0 Motor remain closed position
Button Not Pressed 0	0 Motor remain closed position	0 Motor remain closed position

At first, we used a photocell instead of the IR sensing system. We used environment light as the indicator, so that if the door is open, the photocell sees light and vice versa. However, since the intensity of the environment light matters in this case, we are not able to find a specific reading that always corresponds to open and closed states, which makes it unstable. In fact, the readings sometimes fluctuates in ambient light environment.   Using the IR sensing system solves this problem, since it is emitting and sensing a specific intensity and wavelength of the infrared light, just as how the remote control works to turn on the TV set. From our experiment, this module gives much more accurate readings.

To open the door from the outside, we used an RFID Module from Sparkfun. As the antenna of the module sees a tag of certain frequency , the lock will open. We combined the given code from sparkfun with our previous code of the IR sensing system so that when it sees the tag, the flag variable in the code will be set to 1;otherwise, it the flag will be set to 0. Then we can use the change of this flag variable to control the opening of the lock. We allow ten seconds for the user to push open the door. If the user doesn't do anything, the door will close again.

Code:

```
#include <Servo.h>
Servo myServo;

#include <SoftwareSerial.h>
SoftwareSerial rfid(7, 8);
SoftwareSerial xbee(10, 9);

void check_for_notag(void);
void halt(void);
void parse(void);
void print_serial(void);
void read_serial(void);
void seek(void);
void set_flag(void);

int pos = 0;
int sensorValue;
int buttonValue;
int flag = 0;
int Str1[11];

void setup() {
Serial.begin(9600);
myServo.attach(6);
pinMode(A0,INPUT);  //INPUT from phototransistor
pinMode(A2,INPUT);  //INPUT from AND gate

  // set the data rate for the SoftwareSerial ports
  xbee.begin(9600);
  rfid.begin(19200);
  delay(10);
  halt();
```

```
}

void loop() {
  read_serial();                                  //Check RFID tag
  myServo.write(pos);
  sensorValue= analogRead(A0);   //INPUT from phototransistor
  buttonValue=analogRead(A2);     //INPUT from AND gate

  if(sensorValue>900) //  Phototransistor sees light from IR diode.
    {
      pos = 90;
    myServo.write(pos);

    delay(200);
    }

  if(buttonValue>800) //  Motor in closed state and user pressed button
  {
    pos = 0;
    myServo.write(pos);
    delay(5000);
  }
  if(flag==1)  //User scans the RFID tag
  {
    pos = 0;
    myServo.write(pos);
    delay(10000);

   }

}

/*Below are function calls related to RFID sensing*/
void check_for_notag()
{
  seek();
  delay(10);
  parse();
  set_flag();

  if(flag = 1){
    seek();
    delay(10);
    parse();
  }
}

void halt()
{
 //Halt tag
  rfid.write((uint8_t)255);
  rfid.write((uint8_t)0);
  rfid.write((uint8_t)1);
  rfid.write((uint8_t)147);
  rfid.write((uint8_t)148);
}

void parse()
{
  while(rfid.available()){
    if(rfid.read() == 255){
      for(int i=1;i<11;i++){
        Str1[i]= rfid.read();
      }
    }
  }
}

void print_serial()
{
  if(flag == 1){
    //print to serial port
    Serial.print(Str1[8], HEX);
    Serial.print(Str1[7], HEX);
    Serial.print(Str1[6], HEX);
    Serial.print(Str1[5], HEX);
    Serial.println();
    //print to XBee module
    xbee.print(Str1[8], HEX);
    xbee.print(Str1[7], HEX);
    xbee.print(Str1[6], HEX);
    xbee.print(Str1[5], HEX);
    xbee.println();
    delay(100);
  }
}

void read_serial()
{
  seek();
  delay(10);
  parse();
  set_flag();
  print_serial();
  delay(100);
}

void seek()
{

  //search for RFID tag
  rfid.write((uint8_t)255);
  rfid.write((uint8_t)0);
  rfid.write((uint8_t)1);
  rfid.write((uint8_t)130);
  rfid.write((uint8_t)131);
  delay(10);
}

void set_flag()   //If it sees the tag,flag=1;Otherwise, flag=0
{
  if(Str1[2] == 6){
    flag++;
  }
  if(Str1[2] == 2){
    flag = 0;
  }
}
```

Results

To conclude, our current device works and accomplished many of the features we initially wanted to implement in our device. We did face some problems, which are discussed in the section below.
The device functionality is minimal at this point but it shows that it can be scaled to make a production-grade device. Our device does not work on an actual door at this point. It is a design issue now, not a device issue (main blocks of the system function as expected).

Lastly, this is the first version of our device so design of the device is not completely sketched out on an actual door. In fact, we didn't receive our lock so we couldn't even begin trying our device on a real door. At this point the device seems functional enough but we are not sure about security flaws that could arise from using RFID system.

Problems and Challenges

We got our components very late, so we couldn't understand their functioning properly.However, our device has a lot of issues that need to be resolved. To begin with, some of our components are not working properly. For instance, when our device is in use, the servo motor is constantly vibrating. Additionally, there are a lot of improvements to be made in the way we have wired our device. Our wiring created a lot of difficulties during the first demo of our device.

One way understanding components would directly improve device performance would be through reducing the vibrations of the servo motor (when the device is in use). This could help us save power. Another example is of the Xbee Shield Module. We don't know why we need it but our device does not work without it.

As discussed earlier, it took a long time to figure out how to make a sensing module to work properly with button and the NAND gate. We had to change it from LDR to IR to make it work.

Future Work

Installing it on door would be very difficult with current device configuration. Also, we need to design a casing to protect our device from somebody tampering with it. We have not worked on the casing for our front and back-side platforms at all. A right casing can increase security of our device.

We can even convert our Arduino code into an Arduino library that someone you can use to improve or build their own features on our device. We don't have that functionality with our current code. Also, the code for the RFID detection module is taken from the Sparkfun RFID Eval Module Github Page. We need to encipher our RFID tags as well as integrate their code more properly given enough time.

Sources

[1]"Arduino - Reference", *Arduino.cc*, 2016. [Online]. Available: https://www.arduino.cc/en/Reference/HomePage. [Accessed: 12- Dec- 2016]

[2]A. Weiss, "sparkfun/RFID_Evaluation_Shield", *GitHub*, 2016. [Online]. Available: https://github.com/sparkfun/RFID_Evaluation_Shield/blob/V_1.4/Firmware/RFID_Eval_13_56MHz.ino. [Accessed: 12- Dec- 2016]