EE 459 Final Project

Team 14, Spring 2018

Xin Yu, Matthew Cacho, Liheng Lin

Contents

Appendixes

Problem and Motivation:

Home alarm systems are prevalent in many family houses today. However, there are still some limitations with these systems. For one, there are false intrusions that unnecessarily trouble families at different times of the day. In addition to that, the methods of alarming the user may be annoying at times.

We wanted to develop and design a smart home alarm system to remove the various annoyances that can come with standard alarm systems. Our design tracks the time of day to decide the method of notifying the user. Moreover, our design utilizes sensors that track if a movement may be a real intrusion, or if it's a false alarm. Lastly, our design notifies the user via email if there is an intrusion.

Overview:

We designed this smart home alarm system mainly to detect intrusion. It can be installed in existing homes, instead of having it installed during the construction of the house. The system is composed of two parts, the main control unit and the remote unit. All sensors are on the remote unit, which can be installed in living rooms or bedrooms to detect whether there is an intruder. The main control unit receives information from the remote unit and provides interaction with users. The keypad and LCD screen on the main unit provide information for users and enable users to do settings and other operations. Only one main control unit and one remote unit is required for the system to function. However, our design is also expandable in that the users can purchase more remote units to install in different rooms. To install the system, users first choose different places to install the main and remote unit and then provide power to the units. Users will follow the instruction on the LCD screen to setup a password and the system is ready to work.

Block Diagram:

Prototype:

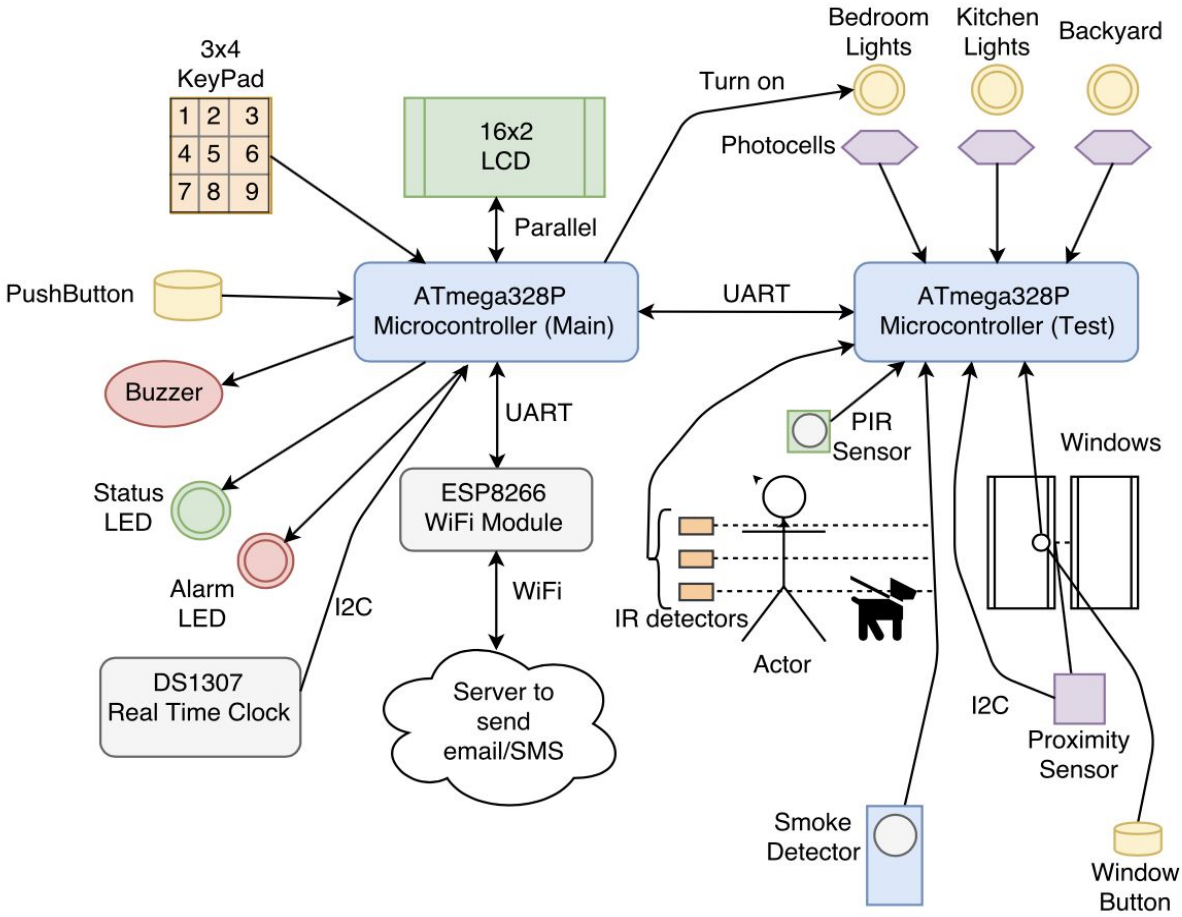


Fig1: Block Diagram from Detailed Design Review

Description of the block diagram:

The main microcontroller takes inputs from the user in terms of keypad and pushbutton, and makes output to the LCD, turn on/off the buzzer or turn on bedroom lights, or send request to the Wi-Fi module which will connect to the server to send emails/SMS when moving objects/window intrusion is detected. A green LED light is used to indicate the system status, and a red LED light is used for the alarm, which lights up if the system detected intrusion. A real time clock is connected to the microcontroller to provide the current time. The Wi-Fi module has serial connection to the main microcontroller to transmit and receive data. It establishes Wi-Fi connection, and connects to server for email/SMS. The main microcontroller has serial communication to the second microcontroller in order to read and analyze inputs from multiple sensors installed on the second board, and control the bedroom light during night when there is a situation that needs to alert the user. Since the ATmega328P microcontroller only has one set of

RX/TX pins, we need a 2-to-1 mux to switch to the serial communication to the Wi-Fi module after the main microcontroller receives alerting situation from the second microcontroller. The RX/TX pins of the main microcontroller listens to the second microcontroller during other time.

The second microcontroller reads inputs from photocells, passive IR sensor, IR detectors, proximity sensor, window button and smoke detector. It analyzes if there is an intrusion/smoke, where intelligence of the alarm system is introduced. Bedrooms, kitchen and backyard are simulated on the second board with another microcontroller. The photocells are installed to determine if the lights in individual rooms are on/off. Photocells are able to differentiate light levels. The combination of passive IR sensors, three sets of IR emitters & detectors detects moving objects and differentiate between human and animals. The three set of IR emitters & detectors are placed at different heights. If the lower two IR emitters & detectors detect blocking objects along with positive feedback from the passive IR sensor, we are able to tell that there is an intruder instead of an small-sized animal. The combination of proximity sensor & window button facing interior indicates if the window is being open from the outside. The combination of smoke detector and photocell for the kitchen light determines if the smoke is generated by cooking or it is a situation that needs to alert the user.

Final Design:

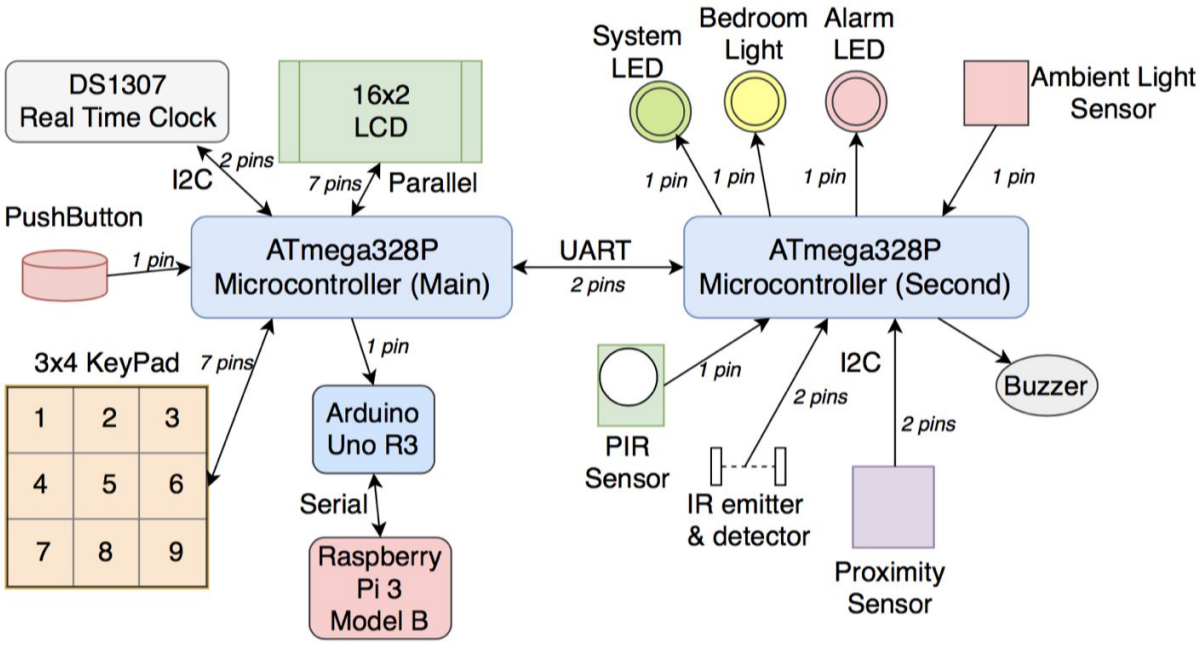


Fig2: Revised Block Diagram After Implemented

Revision to the block diagram during implementation & Reasons:

The definition of the system on/off was ambiguous when we first came up with our design. During implementation, we decides to move the green LED for the system status from the main microcontroller to the second microcontroller. The main microcontroller is always on to wait for inputs from the user in terms of button presses and keypad. The LCD is constantly displaying the current time and the status of the system. By turning the system off, the second microcontroller will no longer be listening to the feedbacks from the sensors & detectors or analyzing the situation.

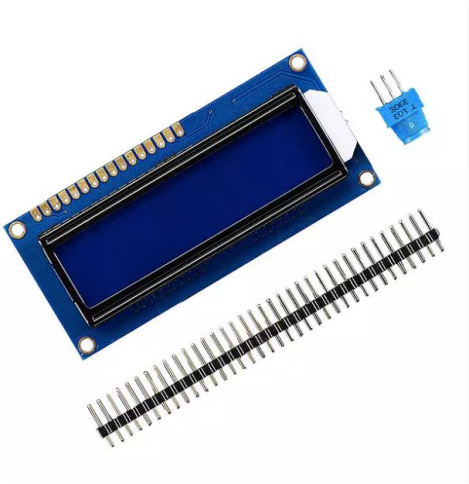
We moved the buzzer from the main microcontroller to the second microcontroller for two reasons. First, we want to alert the user immediately when the second microcontroller sees there is an intrusion, and avoid the situation when the serial communication between the two boards is not working right. Second, during the software implementation, we encountered a difficulty that global variables were not updated in the main while loop. We had to move all the major functions to different interrupts so that the global variables were updated correctly. We managed to let the system status flag updated correctly in the pin interrupt for the pushbutton and the timer interrupt for the real time clock. But we could not get the buzzer on flag reset to 0 after the user set the password correctly in the pin interrupt for the keypad. We believe it is because that we used polling for the keypad and the function for requesting password includes a while loop. Therefore, we had to move the buzzer to the second microcontroller.

For the part to send emails/SMS to the user when there is an alarm, originally we tried to use the Wi-Fi module which is cheaper and more suitable for the scale of our design, but we could only get one-way communication to the Wi-Fi module. In order to get this feature of our project to work, we opted for raspberry pi which operates like a small computer and connects to the wifi. We first used arduino and raspberry pi to get serial communication between ATmega328P and raspberry pi working properly. We added the logic level converter to make up for the voltage difference between Arduino (5V) and raspberry pi (3.3V). After we confirmed that the serial communication was working properly between the two, we connected the raspberry pi directly to the RX/TX pins of the second microcontroller. Because there was not much time left after we got the communication between the raspberry pi and arduino as well as sending emails to the user to work, we decided to connect the arduino and the second microcontroller in order to add the feature in time.

In actual implementation, we did not add all the sensors or detectors as we planned. It would be hard to simulate the smoke in kitchen to include the case of smoke detection. And one of us suggested that we could implement only one set of IR detectors & emitters which would be enough to tell the difference between animals and humans. Our system can include more sensors and detectors to expand for more purposes as long as the pins on the second microcontroller are available. Not only that, but additional microcontrollers for more remote control units could be added, and the arduino is a prime example of that.

Detailed Description of Major Components:

- LCD:



  - The LCD is the main part providing system information for users. It displays the request of new password once the system is installed. After the password is set, it displays current time on the first row and the status of the system on second row during idle time. The request for password will be displayed if users use the pushbutton to turn on or off the system. When users press the '*' to set how much time later for system to be turned on or off, the user input is printed on the screed. If there is intrusion detected, it prints out the specific situation that triggers the alarm, ie: Moving Objects/Window Intrusion. It also prints out "Sending Email" to indicate that a warning is being sent to users through email.

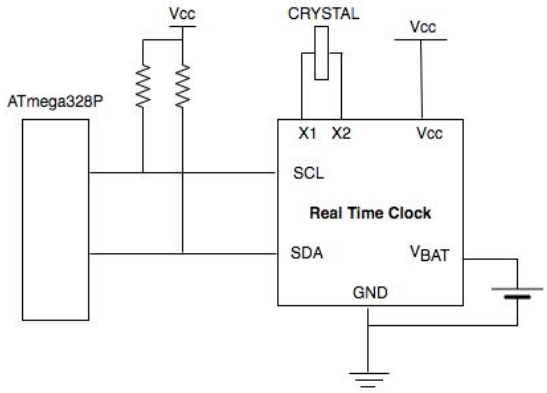
- Keypad:
  - Users use the keypad to input password whenever it is requested. In addition, users can press the "*" key on the keypad to indicate that they want to set the time for the system to be turn on/off.

- Pushbutton:
  - The pushbutton serves as an input for users to turn on/off the alarm system/buzzer/bedroom light. Every time a user presses the pushbutton, password request will be prompted.

- LEDs:
  - The green LED represents the system LED. When the remote unit is on and monitoring, the system LED will be turned on.

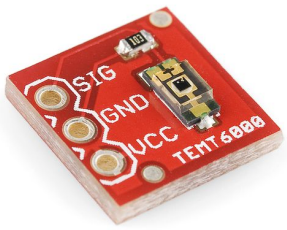
- The red LED represents the alarm LED. When there is moving object detected or the window is opened, the alarm LED will be on indicating intrusion detected.
- The yellow LED represents the bedroom light. During the night time, The bedroom light is turned on to warn the user instead of sounding the buzzer, which prevent the alarm from disturbing neighbors.

- Real Time Clock:



- The real time clock provides time for the main microcontroller through I2C. We sent the current time through I2C to the clock during the first programming, and it will keep updating its time and send back new time through I2C back to the main microcontroller. There is no need to set the time everytime we program the microcontroller. A lithium battery will back up the clock for years when there is power supply for the chip. In this case, users do not need to reset the time after a power outage.

- Ambient Light Sensor:

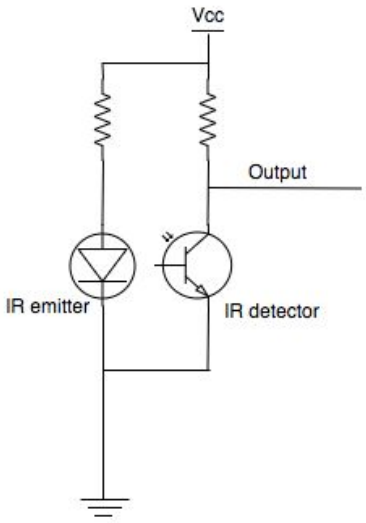


- The ambient light sensor outputs a high signal on its SIG pin when it detects light and low signal when it does not. It is installed on the remote unit to determine whether it is daytime or night.

- PIR Motion Sensor:

  - The PIR motion sensor detects motion by sensing changes in the infrared levels emitted by surrounding objects. The PIR sensor outputs a high signal on its output pin when motion is detected. It detects motion within a range of 30 feet.

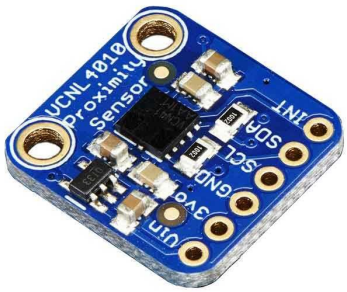
- IR Emitter & Detector:



  - The IR emitter (yellow dot) sends out IR light when it is connect to power and the detector (red dot) receives the IR light. The output shown in the above schematic is high when there is an object that blocks the IR light to the detector. The emitter and detector are installed at a certain height in the house, which filter out small objects such as pets. Once the output signal is high, indicating that some object of enough height is detected, the alarm will be triggered if the PIR motion sensor detects moving object. If the microcontroller receives a low signal from the

emitter & detector, it will not sound the buzzer or turn on bedroom lights even the motion sensor detects movement.

- Proximity Sensor:



  - The proximity sensor measures the distance to other objects within 200mm. The proximity measurement results are stored in its registers. The second microcontroller communicates with it through I2C. The microcontroller first need to write a command starting the measurement, and keeps retrieving results from the proximity sensor. There is no specific description of the 16 bits measurement value, so we first tried to print out the 2 byte on the LCD and found out it could not represent the distance. We then converted the result into decimals and found out that the values went up if some object got closer to the sensor. We use the proximity sensor on the edge of windows. When the window is closed, the distance it measures is very small thus the proximity measurement result is a large number. In the software, we set a threshold and once the result is below the threshold, it indicates that the window is opened.

- Buzzer:
  - The buzzer sounds when there is an intrusion during daytime.

- Main Microcontroller:
  - The main microcontroller takes inputs from a real time clock, a pushbutton and a keypad, and it outputs to the LCD and the Arduino. It obtains current time from the clock and displays it on the LCD. It takes the input from pushbutton to make certain change and displays request for password on the LCD. When the alarm is triggered, the microcontroller will output a high signal to the Arduino, which communicates with the Raspberry Pi to send a warning email to users. The main microcontroller communicates with the second microcontroller through serial

communication. It sends out its current status (system on/off) to the remote unit when it boots up and whenever the status is changed.

- Second Microcontroller:
    - The second microcontroller takes inputs from the ambient light sensor, PIR motion sensor, proximity sensor and the IR emitter & detector. The software, which will be explained in the Software Design section, of the second microcontroller determines whether the alarm should be on according to the input from sensors. If there is intrusion, it turns on the red alarm LED, makes the buzzer sound (during daytime) or turns on the bedroom light (during night). It uses serial communication with the main microcontroller, receives the system status. If the system is on, it turns on the greed system LED and begins monitoring. If the system is off, it turns off the LED and stops all the sensors. When an intrusions is detected, it sends out the information whether it is window being opened or moving objects being sensed to the main microcontroller.

Pictures of Home Alarm System Implemented:

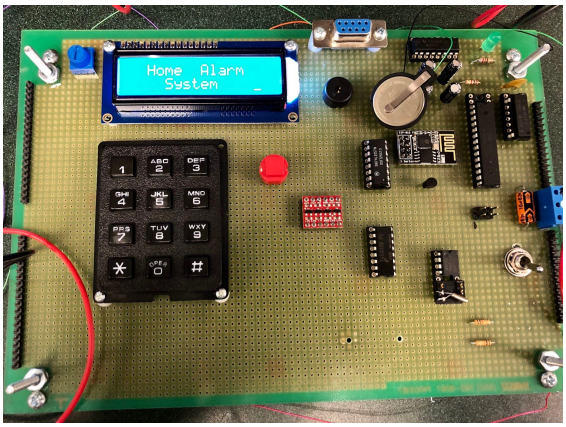
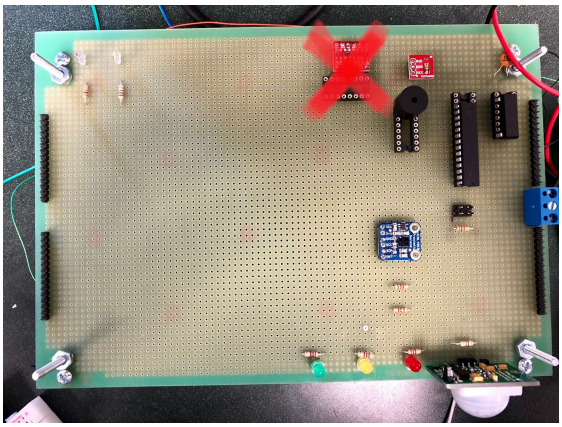


Fig3.(a) The main microcontroller



(b) The second microcontroller

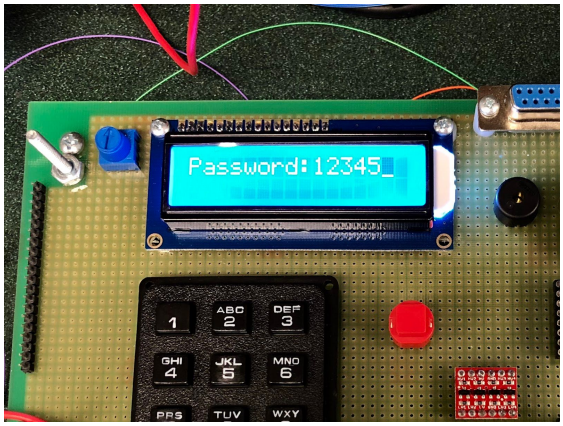
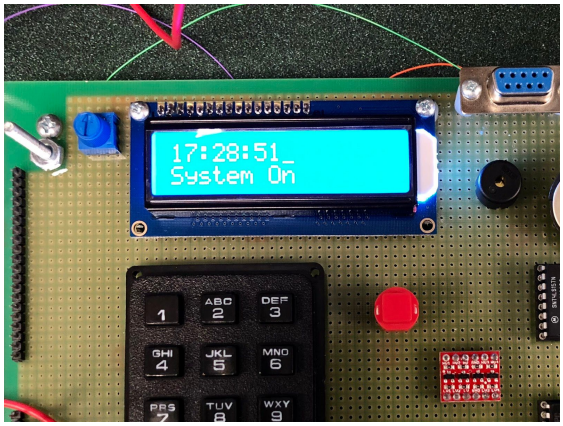
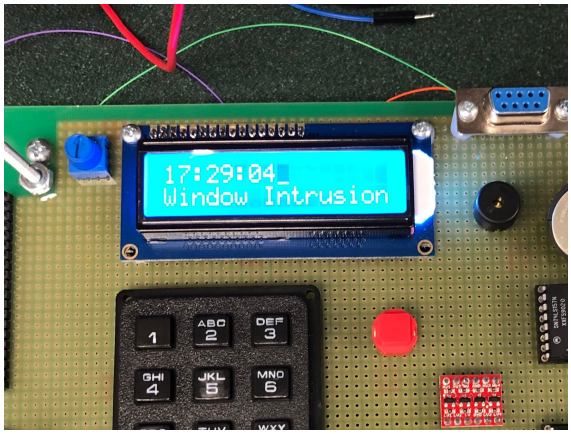


Fig3.(c) User setting password at beginning



(d) System during idle time (system on)



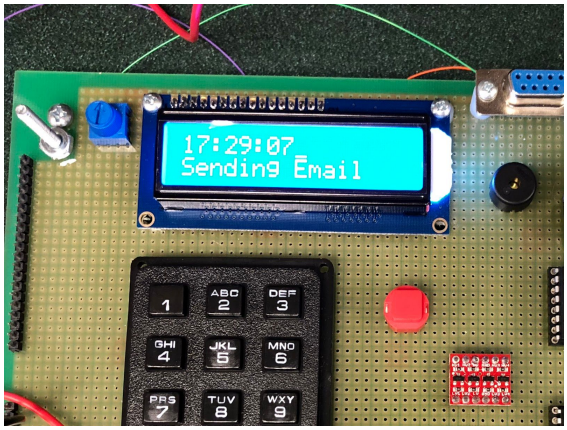
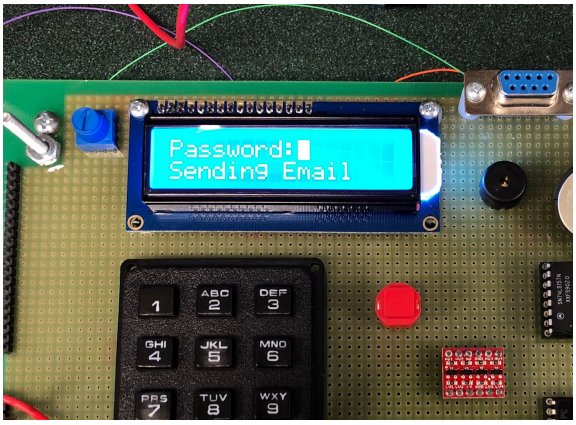


Fig3.(e) Window intrusion detected at the second microcontroller (f) Sending Email



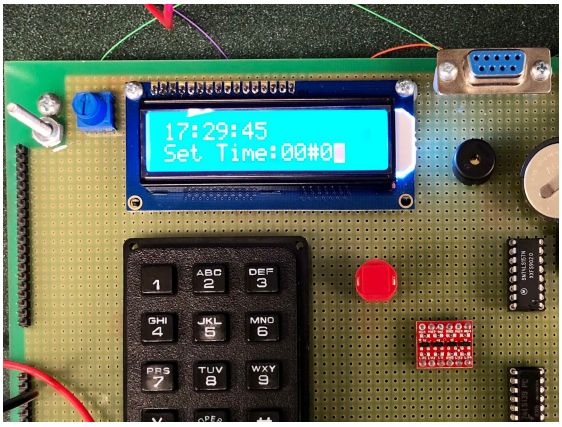
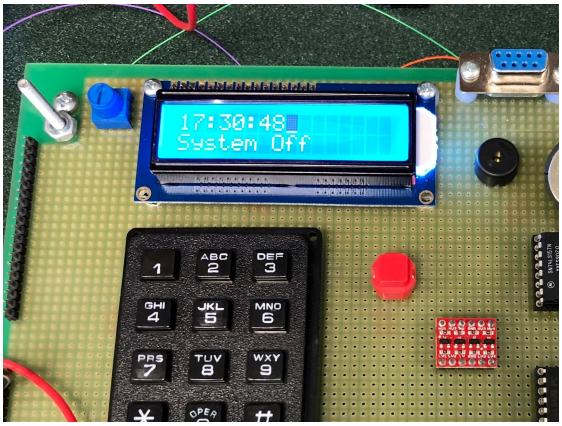


Fig3.(g) Buzzer/Light on at the second microcontroller Password request prompted after user presses pushbutton To turn of the alarm

(h) User presses * on keypad to set time after which the system will be turned off/on



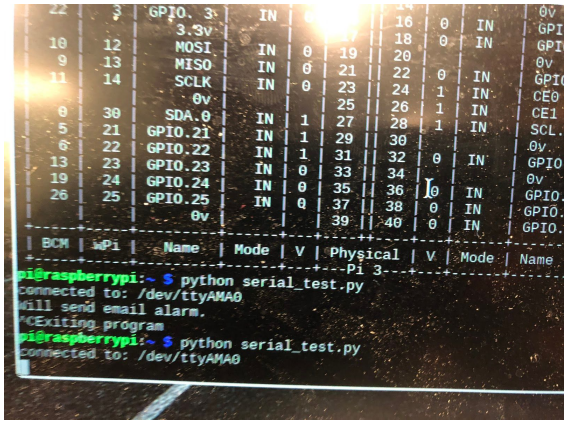


Fig3.(i) System off

(j) Sending email on raspberry pi

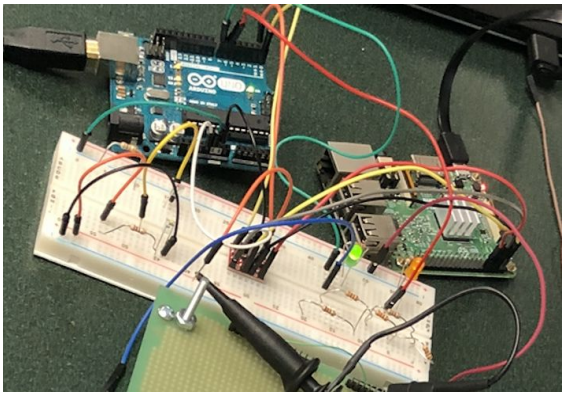


Fig3.(k) Arduino + Raspberry Pi

Software Design:

Section I:

In this section we illustrate the software design both the original one and the implemented one in representation of the state machine so that the flow of operation is more clear. During the actual software implementation, we do not use the state machine principles because we are using different interrupts at different time to flip the system status as well as the alarm status.

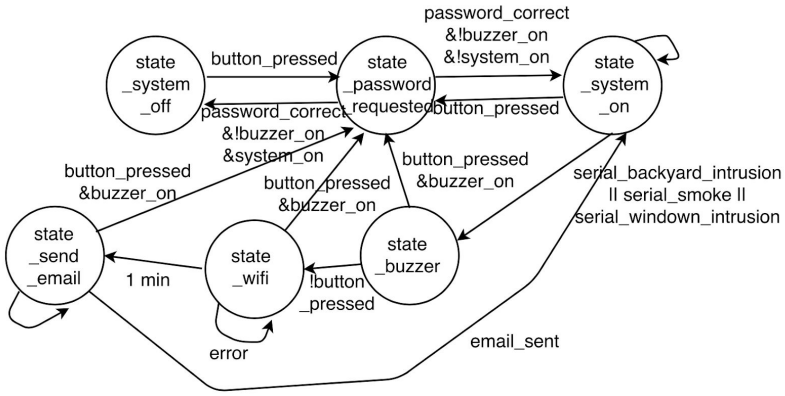


Fig4.(a) State Machine for the main microcontroller from Detailed Design Review

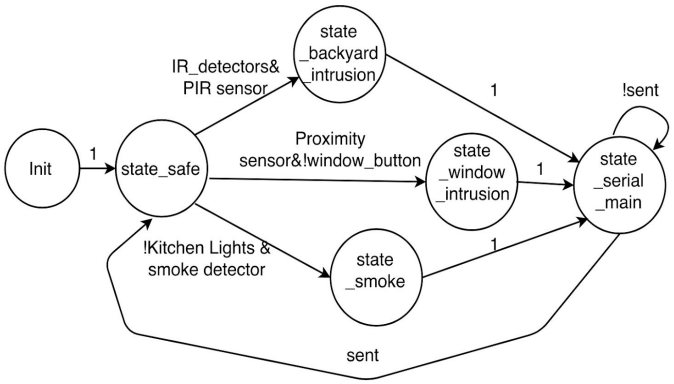


Fig4.(b) State Machine for the second microcontroller from Detailed Design Review

Description of the original state machine:

Our original state machine for the main microcontroller has six states, three of which implements the feature that every time the user tries to modify the status of the alarm system through push button, the user is required to enter the password correctly. In the state system_on, the main microcontroller listens to the second microcontroller through serial communication constantly to receive and determine the kind of intrusion. If intrusion or smoke is detected , the main microcontroller will perform the sequence of operation as follows, turn on the buzzer,

connects to the wifi, and if the buzzer is not turned off for after one minute, a request to send email will be sent to the server by the wifi module.

Our original state machine for the second microcontroller is straightforward. The transition from state_safe to the three different states (state_backyard_intrusion, state_window_intrusion, state_smoke) depends on different combinations of the detectors & sensors output. One goes to any of the states, the transition to state_serial_main is immediate to send alerts to the main microcontroller to turn on the buzzer during daytime or bedroom light during night and display the kind of intrusion on the LCD.

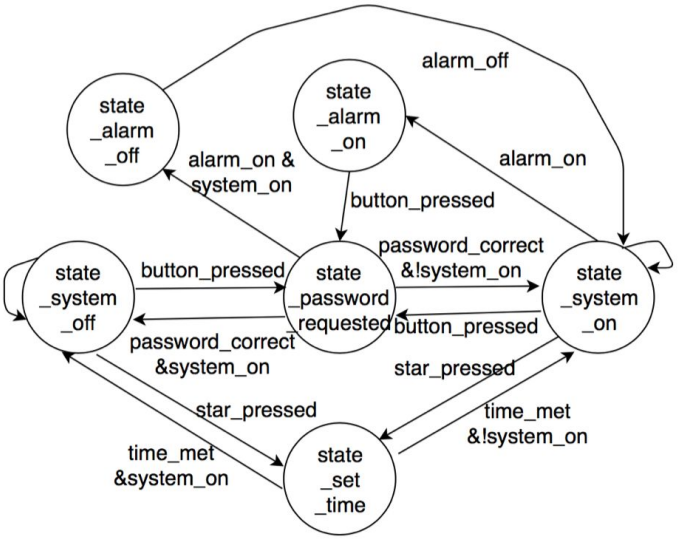


Fig5.(a) State Machine for the main microcontroller from Final Implementation

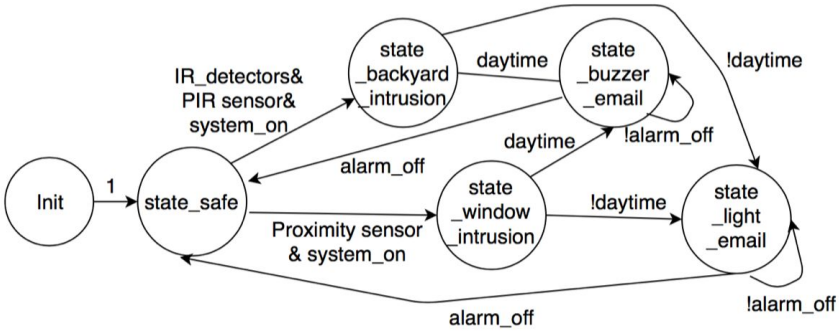


Fig5.(b) State Machine for the second microcontroller from Final Implementation

Description of state machine for the implemented design:

The state machine for the implemented design has the same three states to handle the feature that every time the user tries to modify the status of the alarm system through push button, the user is required to enter the password correctly. The main difference is that we

offload the task to send emails to the user as well as turning on buzzer/bedroom light to the second microcontroller. So everytime, the main microcontroller receives alarm from the second microcontroller, the only task it will do is to validate the user's command to turn off the alarm and ask the second microcontroller to turn off the alarm once the password entered is correct. Additionally, we add the set time feature that the user can turn on/off the system status after 00:00 - 23:59 (hours:minutes) by pressing "*" on the keypad.

The state machine for the second microcontroller is more complicated than the original one. The main difference is that every time it goes to the state_backyard_intrusion or state_window_intrusion, it will turn on the alarm and send a 1 to the arduino which is connected to the raspberry pi to send emails, and meanwhile listen to the main microcontroller for command to turn off the alarm.

Section II:

In this section, we illustrate what interrupts we are using and how they interact to make the system operations coherent.

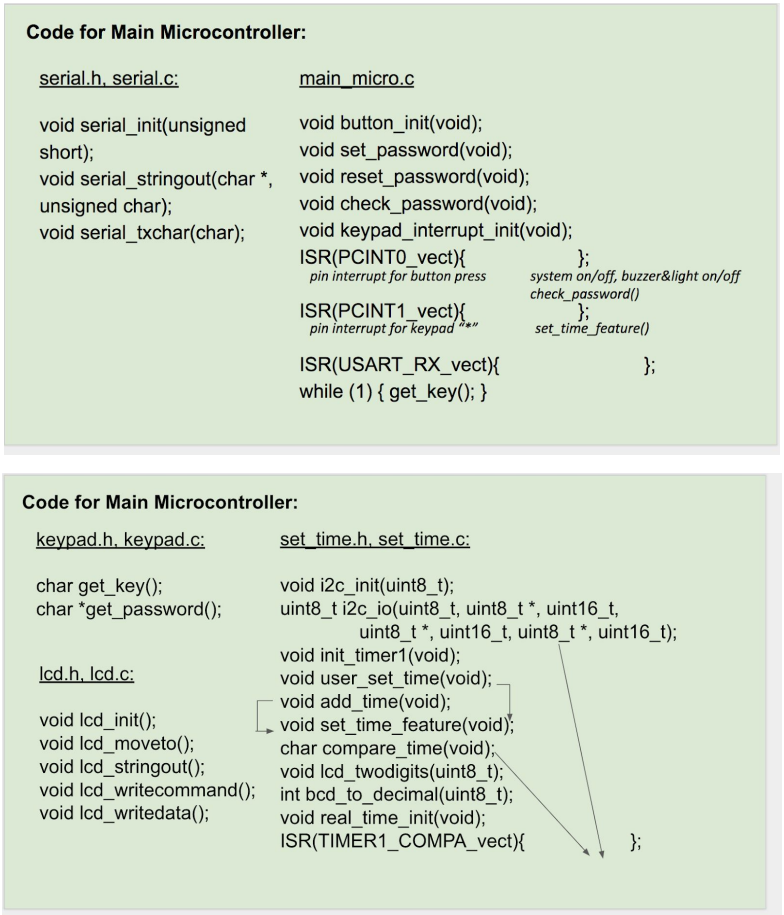


Fig6. Function list for the main microcontroller program

Description of the main microcontroller program:

During the actual software implementation, we encountered difficulties that the global variables used in the interrupts are not updated correctly. So we move all the major functions that flip the system status and alarm status, and flags to display different messages on the LCD at proper time inside different interrupts. For the serial communication between the two microcontroller, we use the RX interrupt to receive messages and a flag is set when the transmission is complete. For the time displaying and set time feature, we use the timer interrupt to retrieve data from the real time clock every one second, and compare the time with the time set by user and flip the system status when the time is met. For the push button, we a pin interrupt to read the button press at the pin, request password from the user, once the password is correct, flip the system status or the alarm system based on the flag updated in the RX interrupt. To trigger set time feature, we use the pin interrupt for the "*" key on the keypad. Inside the main loop, we only call the function to get key presses on the keypad.

The Atmeg328p chip has 32kB of flash memory. As indicated in Fig7, our program for the main microcontroller costs 6406 bytes, which is 19.5% of the flash memory. So there are about 80% of the flash memory left for additional features.

```
rm -f main.hex
avr-objcopy -j .text -j .data -O ihex main.elf main.hex
avr-size --format=avr --mcu=atmega328p main.elf
AVR Memory Usage
----------------
Device: atmega328p

Program:    6406 bytes (19.5% Full)
(.text + .data + .bootloader)

Data:        257 bytes (12.5% Full)
(.data + .bss + .noinit)

avrdude -c usbtiny -P usb -p atmega328p -U flash:w:main.hex:i

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e950f
avrdude: NOTE: "flash" memory has been specified, an erase cycle will be performed
         To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "main.hex"
avrdude: writing flash (6406 bytes):

Writing | ################################################## | 100% 8.29s

avrdude: 6406 bytes of flash written
avrdude: verifying flash memory against main.hex:
avrdude: load data flash data from input file main.hex:
avrdude: input file main.hex contains 6406 bytes
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 6.17s

avrdude: verifying ...
avrdude: 6406 bytes of flash verified

avrdude: safemode: Fuses OK (H:07, E:D9, L:E0)

avrdude done.  Thank you.
```

Fig7. Screenshot of terminal while programming the main microcontroller

Section III:

In this section, we illustrate the operations between arduino and raspberry pi, and their connection to the second microcontroller.

The arduino's RX pin is connected to PC1 of the remote unit, while the arduino's TX pin is connected to the serial communication pin of the raspberry pi. A logic level converter regulates the voltage between the arduino and the raspberry pi, since the raspberry pi can only receive 3.3 volts and the arduino runs on 5 volts. If an intrusion is detected in the remote unit, a signal is sent to the arduino. The arduino then sends a high signal to the raspberry pi.

```
ser = serial.Serial(
    port='/dev/ttyAMA0', baudrate=9600, timeout=1
    )
time.sleep(2.0)
ser.flushInput()
print("connected to: " + ser.portstr)
try:
    i = 0
    while 1:

        i+=1

        if ser.inWaiting() > 0:

            response = ser.read(1)
            time.sleep(.5)
            print("Will send email alarm.")

            server = smtplib.SMTP('smtp.gmail.com',port=587)
            server.starttls()
            server.login(fromaddr,'459capstone')

            text = msg.as_string()

            server.sendmail(fromaddr, toaddr, text)
            server.quit()

            time.sleep(5)
            print("Email sent.")
            break

except KeyboardInterrupt:
    ser.close()
    print("Exiting program")
```

Fig8. Python script in raspberry pi

The script in the raspberry pi runs on a loop, waiting for a signal from the arduino. Once a signal is detected, the raspberry pi connects with an smtp server (in our case, Gmail) and formats a message specific to the intrusion. After an email has been sent, the python script will wait once again for a new signal from the arduino. The python script stops running when the user exits the program, or the arduino is shut down. The arduino device only sends one signal for every intrusion detected.

Challenges Encountered:

One problem we encountered during our design process was implementing the ESP8266 Wi-Fi module, and making it work with our system. In the beginning, many documentations stated that the Wi-Fi module only works with a 3.3 voltage power supply. We used a voltage regulator and a logic level converter so that the Wi-Fi module receives a 3.3 voltage from all signals, and that any signal it returns gets converted to 5 volts. The Wi-Fi module's RX pin was connected to the TX pin of the main unit, and the Wi-Fi module's TX pin was connected to a mux. One of the mux's output pins was connected to the RX pin of the main unit. The mux regulated if the main unit was receiving data from the Wi-Fi, or the remote unit.

Once this was setup, we connected the pin of the Wi-Fi module as dictated in it's schematics. However, it wouldn't work and receive any signals. We realized two problems. First, one of the pinouts of the Wi-Fi module, called CH_PD, had to be pulled-up to VCC during setup. This allows users to send the AT command to the Wi-Fi module. Second, the logic level converter regulated the voltage received by the Wi-Fi module to 3.3 volts; however, it wasn't receiving enough current so it couldn't detect any data. So instead of using the logic level converter, we connected it directly to an external power supply.

We used a laptop to connect directly with the module and test if it was receiving any data. To open a terminal and send commands, we used Putty. We were able to see that AT commands were being sent to the Wi-Fi module. However, having the Wi-Fi module send back signals proved to be a challenge. Not only that, but connecting the Wi-Fi module directly to the main microcontroller proved to be another challenge. Initially, when we sent AT commands from the serial link, the Wi-Fi module could not detect it. When we were able to make it work, we had a limited time to figure out how to make the Wi-Fi module respond with its own signal and send it back to the main unit. Not only that, finding a way for the module to connect with a SMTP server and format an email to send to the user proved to be an obstacle. We then decided to switch to using a raspberry pi, since setting it up was not as difficult.

Future Improvement:

Of course, our design have potential for improvement. As mentioned before, the arduino connected to the remote unit could be removed in favor of simply connecting the raspberry pi directly to the remote unit microcontroller. Because of time constraints, we opted not to do this. However, combining the arduino with the remote unit microcontroller is definitely feasible.

Not only that, but more microcontrollers can be connected to the main unit. Just like the arduino connected to the remote unit, a similar approach could be designed. This allows for the expansion of utilities, such as smoke detection, water leakage, garage sensors, and much more.

Also, even though the Wi-Fi module proved to be difficult to use, if we were given more time and knowledge of the device, a way to implement it would have surfaced. However, using the raspberry pi also adds more benefits. The raspberry pi has the advantage of having bigger storage because of its SD card. This gives more potential uses, such as taking photos or videos of intruders, and saving them for future use. Another thing that could be implemented in the pi is saving a log of every intrusion that the system detects. This would be good if the user wants to see any history of intrusion.

We also considered wireless communication between the main unit and the remote unit. The serial connection could be replaced with an RF transmitter and receiver module. This eliminates the long and messy wires that plague the current design.

Lastly, there are more ways of notifying the user of intrusion. There are many GSM and GPRS modules in the market that allow for SMS communications. Although most of these modules are expensive, it gives the option of notifying the user without relying on internet access. Not only that, but it provides the opportunity of possibly calling 911 in case of very urgent emergencies.

There are many possibilities with our current design, and given more time, the system is expandable for more opportunities.

Conclusion:

In conclusion, our design opens the way for smarter alarm systems. By smartly ruling out false intrusions, and providing various ways of notifying the user, our design fits nicely for many homes today. The system also checks and tracks the time of day, and this alters the way the system will communicate with the user. This removes any annoyances that an alarm noise would do during the night, or unnecessary electricity consumption during the day. It also allows for further expansion, so more areas in the house could be monitored. Ultimately, our design aims to provide an easy and accessible home alarm system that users would find simple and convenient to use.

Appendix:

Parts List:

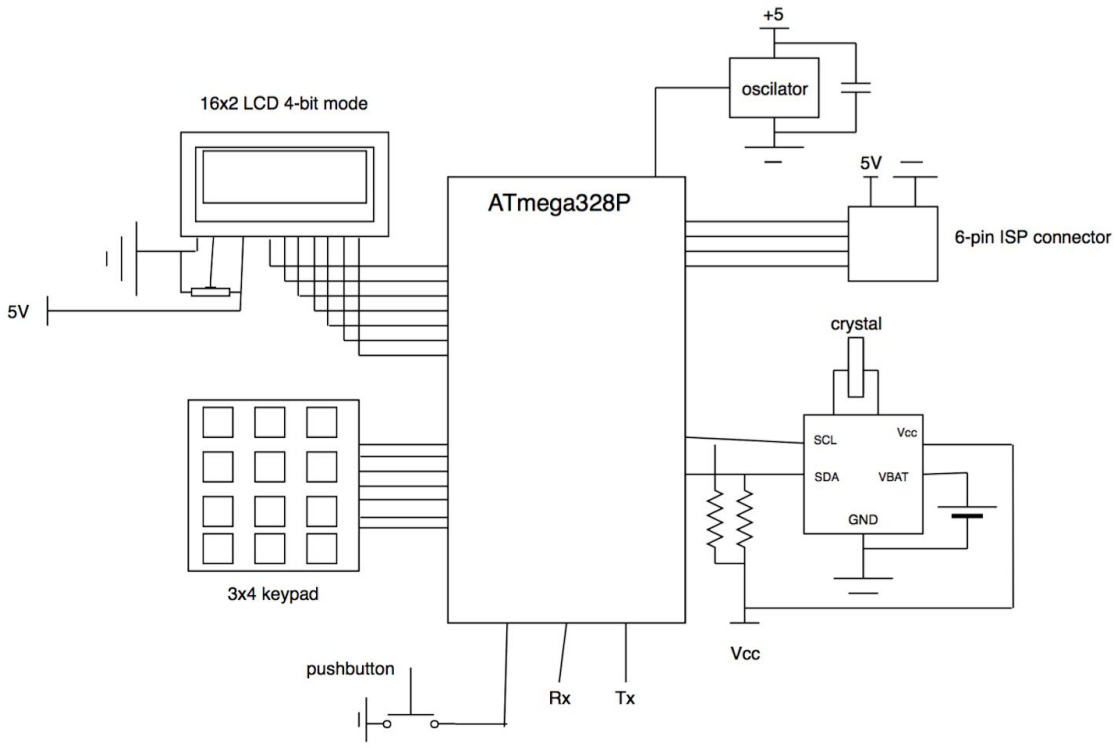
Brand:	Item:	Quantity:
Atmel ATmega328p	Microcontroller	2
Parallax 555-28027	PIR motion sensor	1
Sparkfun SEN-00241	IR emitter and detector	1
Adafruit 466	VCNL4010 light sensor	1
Sparkfun BOB-08688	Ambient light sensor	1
Digikey 1528-1502-ND	LCD, 16x2, white on blue	1
Digikey 1528-2161-ND	4x3 keypad	1
Digikey 401-1966	Button, red	1
Sparkfun BOB-12009	Logic level converter, 4 bits	1
Adafruit 160	Buzzer,12mm, 3-30V, square-wave input	1
	Yellow, Red, and Green LEDs	3
DS1307	Real-time clock	1
	Raspberry Pi 3 Model B	1
	Arduino Uno R3	1

Detailed Cost List:

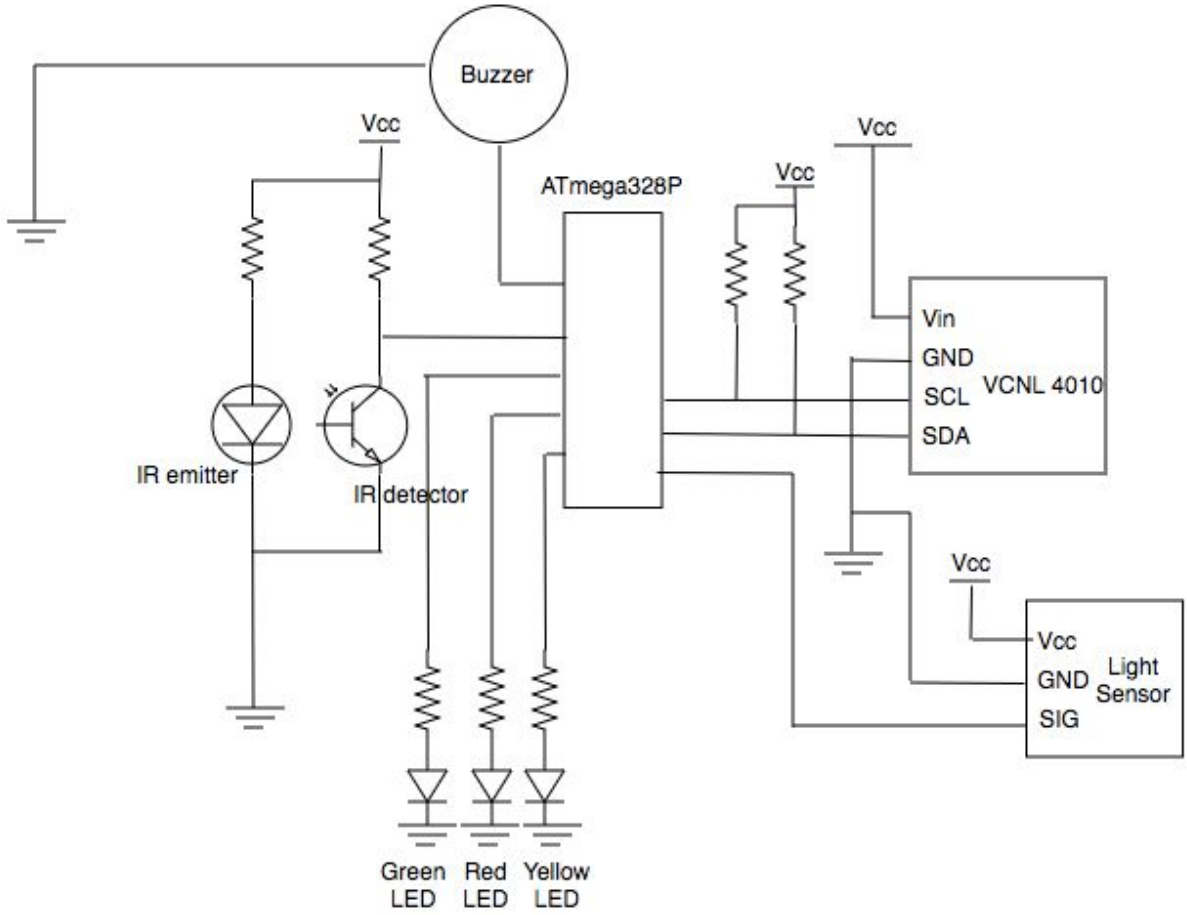
Item:	**Price:**
Microcontroller	$2.59 x 2
PIR motion sensor	$14.99
IR emitter and detector	$1.95
VCNL4010 light sensor	$7.50
Ambient light sensor	$3.95
LCD, 16x2, white on blue	$9.95
4x3 keypad	$4.95
Button, red	$0.50
Buzzer,12mm, 3-30V, square-wave input	$1.50
Yellow, Red, and Green LEDs	$0.40 x 3
Logic level converter	$2.95
Real-time clock	$3.37
Raspberry Pi 3 Model B	$35.00
Arduino Uno R3	$22.00
Total Cost:	$114.99

Schematics:

Main Microcontroller:



Second Microcontroller:



Pin Layout:

Main Microcontroller:

Main Micro Controller				
PB0	14	~~buzzer~~		
PB1	15	button		
PB5	19	~~MuxSelect~~		
PD2	4	Col3	3	
PD3	5	Row1	4	
PD4	6	DB4	11	LCD-DATA
PD5	11	DB5	12	LCD-DATA
PD6	12	DB6	13	LCD-DATA
PD7	13	DB7	14	LCD-DATA
PB4	18	RS	4	LCD-RS
PB3	17	RW	5	LCD-RW
PB2	16	E	6	LCD-E
PD0	2	~~Mux PinY~~	*PD1-Test*	RX
PD1	3	~~WiFI RX~~	*PD0-Test*	TX
PB7	10	Row2	5	
PC0	23	Row3	6	
PC1	24	Row4	7	
PC2	25	Col1	1	
PC3	26	Col2	2	
PC4	27	I2C	SDA	Real Time
PC5	28	I2C	SCL	Real Time

Second Microcontroller:

PB0	14	IR output		
PB1	15	light sensor output		
PB2	16			
PD2	4	Vout	4	PIR-Output
PD3	5	LED TEST		LED-Test
PD4	6	LED for lights		LED for lights
PD5	11			
PD6	12	system LED		
PD7	13	buzzer		
PB4	18			
PB3	17			
PB5	19			
PD0	2		*PD1-Main*	RX
PD1	3	~~Mux_PinA~~	*PD0-Main*	TX
PB7	10			
PC0	23			
PC1	24	Arduino		
PC2	25			
PC3	26			
PC4	27	I2C	SDA	
PC5	28	I2C	SCL	

Signature Sheet:

	Matthew Cacho	Liheng Lin	Xin Yu
System Design	33%	33%	33%
Hardware Design	33%	33%	33%
Hardware Assembly	50%	50%	0%
Hardware Debugging & Testing (w/o WiFi)	0%	50%	50%
Microcontroller Software Design (w/o WiFi)	0%	10%	90%
Microcontroller Software Debugging (w/o WiFi)	10%	30%	60%
Wi-Fi Module & Raspberry Pi (Hardware & Software)	95%	0%	5%
System Integration	33%	33%	33%
Project Proposal	0%	50%	50%
Detailed Design Review Presentation	30%	30%	40%
Final Presentation	33%	33%	33%
Final Report	33%	22%	45%

Signature:_____

Signature:_____

Signature:_____