Motion Sensor Alarm System (MSAS)

Abubakr Abdulgadir, Cody Jenkins, Farnoud Farahmand

12/02/2014

Contents

1.0 – Introduction 4
1.1 – Motivation 4
1.2 – Problem Description 4
2.0 – System Description 4
2.1 – User Work Flow 4
2.2 – System State Description 5
3.0 – System Component Description 6
3.1 – Keypad Description 6
3.1.1 – Hardware Description 6
3.1.2 – Software Description 6
3.2 – LCD Description 7
3.2.1 – Hardware Description 7
3.2.2 – Software Description 8
3.3 – Motion Sensor Description 8
3.3.1 – Hardware Description 8
3.3.2 – Software Description 9
3.4 – Speaker Description 9
3.4.1 – Hardware Description 9
3.4.2 – Software Description 10
4.0 – Results and Conclusions 10
Appendix A – Team Members 11
Appendix B – Parts List 12
Appendix C – System Schematic 13
Works Cited 14

Table of Figures

Figure 1.0 - System Flow 4
Figure 2.0- MSAS System State 5
Figure 3.0-Keypad to MSP430 Connection 6
Figure 4.0-getchr() function 7
Figure 5.0-LCD to MSP430 Connection 7
Figure 6.0-Motion Sensor to MSP430 Connection 9
Figure 7.0-Speaker to MSP430 Connection 10
Figure 8.0-System Schematic 13

1.0 – Introduction

1.1 – Motivation

Our Motion Sensor Alarm System (MSAS) was inspired by a previous ECE 511 semester project that involved a system that secured doors by knocking in a specific pattern. When the pattern was correct, the door would unlock. This simple security system inspired us to think of other simple and practical security systems.

1.2 – Problem Description

The problem is to design and build a simple and cost effective motion sensor alarm system that can be armed and disarmed via a 4 digit Personal Identification Number (PIN).

2.0 – System Description

2.1 – User Work Flow

Figure 1.0 shows the MSAS's high level process flow from a user's perspective.

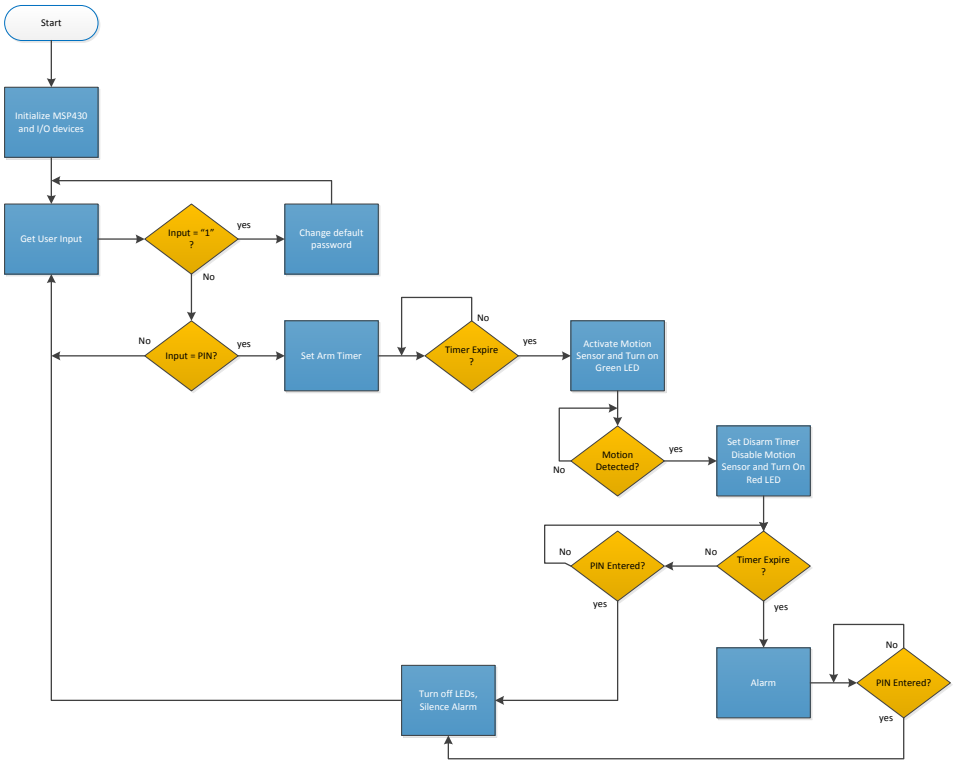

Figure 1.0 - System Flow

The user would first place the MSAS in a location they wish to monitor activity. The user would then turn on the MSAS via the battery pack located inside the MSAS housing. The user is then prompted for the default 4 digit PIN, which is the character sequence "1234". If the user does not want to use the default PIN, they have the option to change the default password by pressing the "1" character followed by pressing the "#" character. The

user would then enter a 4 digit PIN and press the "#" key to submit the new PIN. If the user enters too few characters during the arming process (i.e. less than 4), the system informs the user via the LCD screen that too few characters have been entered. The user also has the option to clear a password if they make a mistake during the arming process by pressing the "*" key. This clears the PIN value and the characters printed on the LCD screen. Once the user's PIN is submitted, the MSAS starts a 10 second timer to allow the user time to get away from the MSAS. After the 10 second timer expires, the motion sensor is enabled, the green LED lights up and the MSP goes into low power mode.

The MSAS's motion sensor continues monitoring the area in front on it until it detects movement. Once movement is detected, the motion sensor is disabled, a 10 second timer is triggered and the red LED lights up. This 10 second timer allows the user to disable the MSAS before the alarm sounds. A user would enter their previously entered 4 digit PIN to disable the MSAS. If the user's PIN was entered successfully, the MSAS would go into the disarmed state. If the user's PIN is not entered successfully before the 10 second timer expires, the alarm sounds and the red LED blinks. The user would then have to enter their PIN to turn off the alarm and successfully disarm the MSAS. The user would then have to rearm the system via their initially entered PIN or they have the option to create a new PIN by the previously described method.

2.2 – System State Description

This section describes the previously discussed user process flow into more detail via a system state diagram. **Figure 2.0** shows the system state of the MSAS.

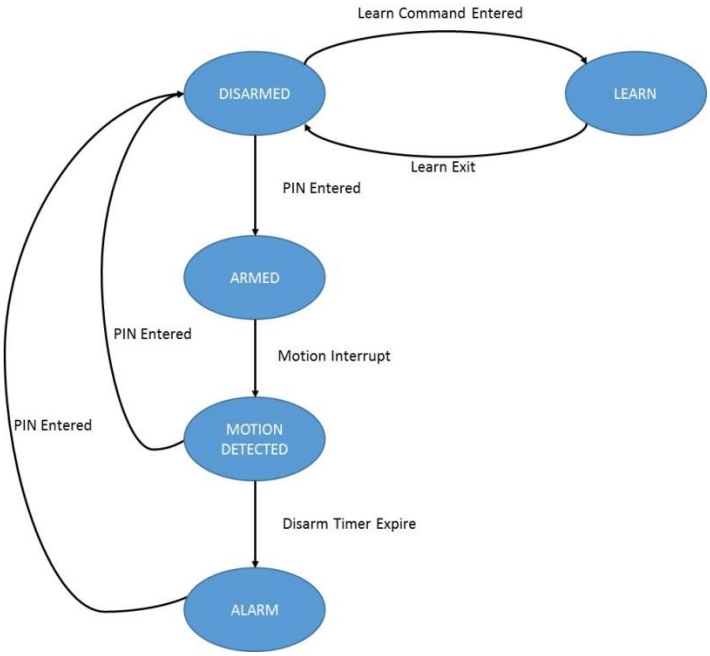

Figure 2.0- MSAS System State

When the MSAS is first turned on, it starts out in the DISARMED state. This state calls a function to get four characters for the PIN from the keypad. After four characters have been pressed, the "#" character is expected to enter the PIN and transition to the next state. If four characters are not entered, and the "#" character is pressed, the state does not transition and remains in the DISARMED state. From the DISARMED state, the LEARN and ARMED state can be reached. The LEARN state is reached by entering a specific character sequence during the DISARM state. If the sequence "1#" is entered, the LEARN state is entered. The LEARN state allows for a new PIN to

be entered. Once the LEARN state is set, the state is exited and returns to the DISARM state so that a PIN can be entered. Once a PIN has been entered, the state gets set to ARMED. The ARMED state enables the motion sensor, sets the green LED, and the interrupt on port 1. The system will remain in low power mode with the general interrupt flag enabled. When the motion sensor detects motion, the port 1 interrupt service routine is triggered and the ARMED state is transitioned to the MOTION DETECTED state. The MOTION DETECTED first disables the motion sensor for power preservation purposes. The state flag is then set to MOTION_DETECTED and the red LED is set. A 10 second timer is started and the interrupt service routine is exited. The state can then transition to either the DISARMED state or the ALARM state based on if the timer expires or the PIN is correctly entered. If the PIN is correctly entered, the state is set to DISARMED and the MSAS can be rearmed. If the timer expires before the pin is entered, the state is transitioned to the ALARM state. The ALARM state triggers the speaker to sound the alarm. The ALARM state can only be transitioned to the DISARMED state via a correctly entered PIN.

3.0 – System Component Description

The following subsections describe in detail how each system component was implemented with respect to hardware and software.

3.1 – Keypad Description

This section describes in detail how the keypad was implemented with respect to hardware and software. The keypad is used as the primary component to arm and disarm the MSAS.

3.1.1 – Hardware Description

Figure 3.0 shows how the keypad was implemented in hardware with the MSP430F5529 microcontroller.

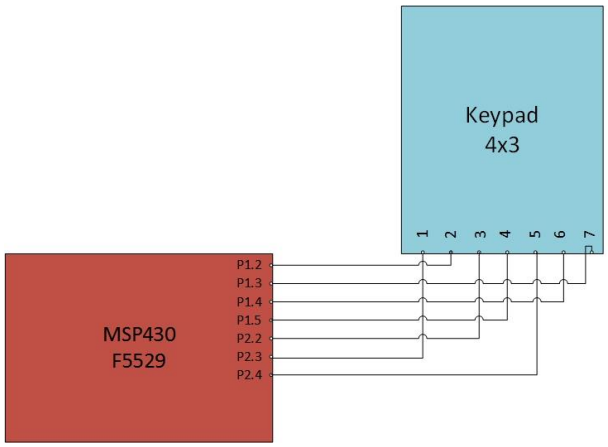

Figure 3.0-Keypad to MSP430 Connection

The keypad has a total of seven pins where pins 2, 4, 6, and 7 are used for the rows and pins 3, 1, and 5 are used for the columns. The columns use the internal pull-down resistors on the pins to keep the columns logic. When a key is pressed, the corresponding column will be set to logic high if high logic appears on the corresponding row. After the column reads logic high, the columns can be read to get the correct key value. The key scanning and decoding functions will be discussed in greater detail in the following software description section.

3.1.2 – Software Description

Figure 4.0 below shows the software flow of the *getchr()*function that is used to return the key values when a key is pressed on the keypad.

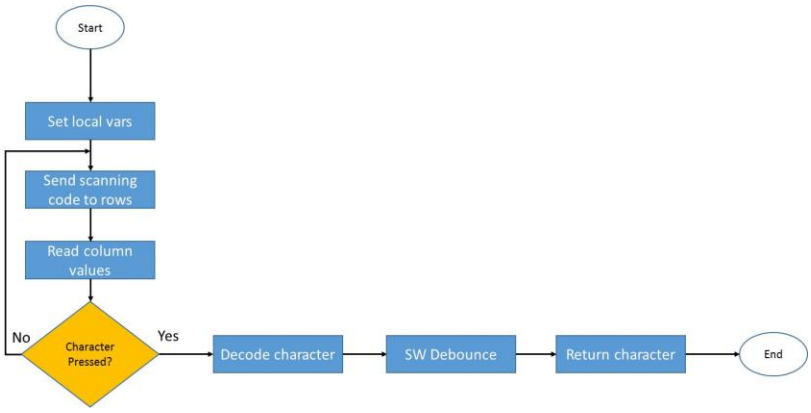

Figure 4.0-getchr() function

The *getchr()* function starts off by setting local variables used to scan the rows and read the columns of the keypad. The function then enters a for-loop that writes values to the rows and reads from the columns of the keypad. If a character is pressed, switch case logic finds the matching row/column pair by looking for logic high values and gets the key value. After the key has been found, a small software debounce loop is executed and the key is returned. If a character is not pressed, then the function continues to scan the rows and read the columns of the keypad. The *getchr()* function is used by another function called *get_input()* that stores four characters for the PIN. The characters that are returned by the *getchr()* function are also used to display the value on the LCD screen.

3.2 – LCD Description

This section describes in detail how the LCD was implemented with respect to hardware and software. The LCD is used to display the PIN value and various states of the MSAS.

3.2.1 – Hardware Description

Figure 5.0 shows how the LCD interfaces with the MSP430F5529.

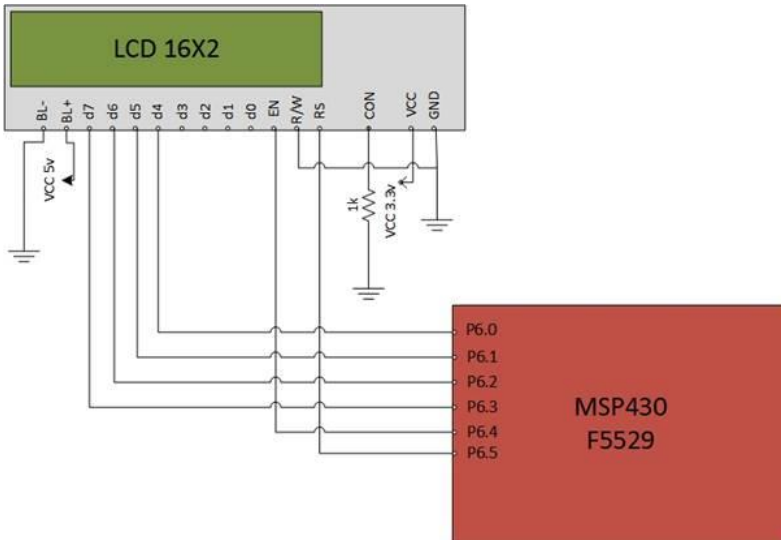

Figure 5.0-LCD to MSP430 Connection

The LCD is a 16 character by 2 line display that runs at 3.3V and utilizes the common ST7066/HD44780 parallel interface. Interface code is widely available for many different controllers and systems. 7 or 11 general I/O pins are needed to interface with this LCD screen. It includes a yellow LED backlight. A 1K ohm resistor was used to

adjust the contrast of the LCD so characters printed to the screen can be visible. **Table 1.0** describes each pin of the LCD.

Pin no.	Symbol	External Connection	Function
1	V_{SS}	Power Supply	Signal ground for LCM
2	V_{DD}		Power Supply for logic for LCM
3	V_0		Contrast Adjustment
4	RS	MPU	Register Select Signal
5	R/W	MPU	Read/Write Select Signal
6	E	MPU	Operation (data read/write) enable signal
7~10	DB0~DB3	MPU	Four low order bi-directional three-state data bus lines. Used for data transfer between the MPU and the LCM. These four are not used during 4-bit operation.
11~14	DB4~DB7	MPU	Four high order bi-direction three-state data bus lines. Used for data transfer between the MPU
15	LED+	LED BKL Power Supply	Power Supply for BKL
16	LED-		Power Supply for BKL

Table 1.0-LCD Pin Mapping

Pins 7 through 10 not used due to the use of 4-bit operation mode. The backlight of the LCD is powered using 5.0v input.

3.2.2 – Software Description

An open source library was used for the ST7066 LCD controller [1]. The ports and timers in the open source library were modified to make it work with the MSP430F5529. To initialize the LCD, the *void lcdInit ( )* function is provided and is called in the first stages of the main function during initialization of the MSP430. To print characters to the LCD's screen, the *lcdSetText (char*, int, int)* function is provided. The first argument of the method is the character to be printed, and the second and third arguments are the position on the screen. The second argument is for the columns (0-15) and the third argument is for the rows(0 or 1). The third and final function leveraged from the open source library is *void lcdClear ( )*. This function is called when the "*" key is pressed and when states change within the code (i.e. arm to disarm, PIN set, etc…).

3.3 – Motion Sensor Description

This section describes in detail how the Motion Sensor was implemented with respect to hardware and software. The motion sensor used is a PIR (Passive Infrared Motion Detector). It works by receiving infrared emitted form objects like humans and animals. The sensor detects sudden changes in the infrared energy it receives.

3.3.1 – Hardware Description

Figure 6.0 below shows how the motion sensor was implemented in hardware with the MSP430F5529 microcontroller.

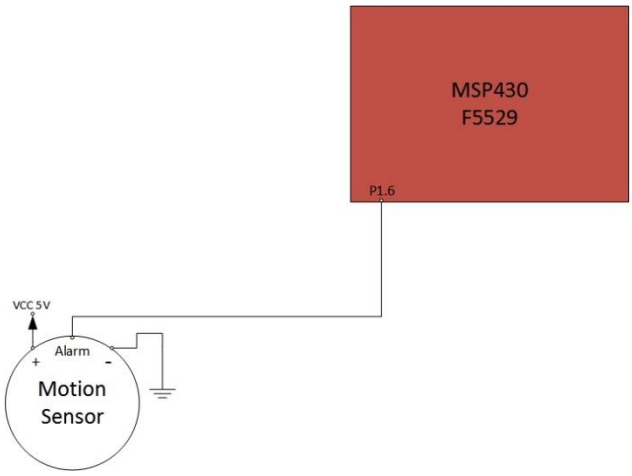

Figure 6.0-Motion Sensor to MSP430 Connection

The sensor used needs 5 V to operate. It needs 1-2 seconds to capture a snapshot of the still room, then when it detects motion, the alarm pin goes low. The motion sensor is connected to pin 1.6. The internal pull-up resistor is used to pull the sensor pin high. When motion is detected by the sensor, the pin reads logic low.

3.3.2 – Software Description

The function *initMotionSensor()* is responsible for initializing the motion sensor by setting port direction, internal pull-up resistor, and interrupt settings.The *enableMotionSensor()* is used to enable the motion sensor interrupt. This function sets a timer to expire in about 2 seconds to allow the sensor to capture a snapshot of the still room. When the timer expires it calls the *doEnableMotionSensor()* which enables the interrupt on the high/low transition. When the interrupt is triggered, the service routine changes the state to MOTION_DETECTED and sets a 10 seconds timer to allow a correct PIN to be entered. If the PIN was not entered within this time frame the alarm sounds.

3.4 – Speaker Description

This section describes in detail how the Speaker was implemented with respect to hardware and software. The speaker is used to generate the alarm sound.

3.4.1 – Hardware Description

Figure 7.0 below shows how the speaker was implemented in hardware with the MSP430F5529 microcontroller.

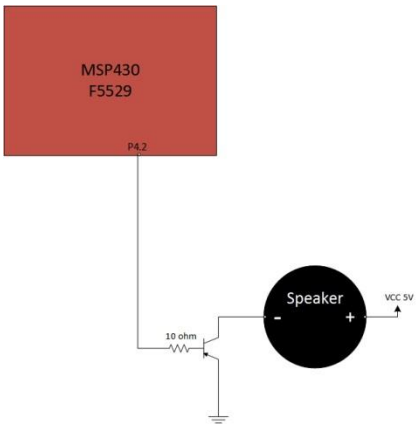

Figure 7.0-Speaker to MSP430 Connection

The speaker is connected to the MSP430 using a transistor to drive the speaker. The base of the transistor is connected to pin 4.2 on the MSP430. A square wave is used to generate sound in the audible frequency.

3.4.2 – Software Description

Timer A2 channel 0 is used to toggle the pin connected to the speaker. The timer is configured in continuous mode. Each time capture/compare channel 0 interrupt is triggered, pin 4.2 is toggled, generating an audible tone. Timer A2 channel 1 is used to generate a repetitive tone-silence pattern in the alarm. For the rest of the system software these details are hidden by two functions used activated/silence the alarm. The *alarm()* function activates the alarm by setting up the timer and compare channels as well as enabling their interrupts. The *shutup()* function works by disabling those interrupts.

4.0 – Results and Conclusions

All components were successfully completed, tested and integrated into one logical unit. The code for the keypad scans, decodes, and returns the correct key when pressed. The LCD code correctly displays the text when called to do so. The Speaker code correctly sends a square wave form via a timer to the speaker to generate a loud sound when called in code. The motion sensor code correctly handles events when motion is detected. Overall, all targeted functionality was completed on time. The first lesson learned was big picture thinking. We often got caught up in the fine details of the functionality of the project and neglected some of the big picture features until the last two weeks of the project. We would have had an easier time completing functionality if we had thought about the big picture of the project more during the beginning of the class. Another lesson learned was time management. We spent a good amount of time ordering and wiring the components when we should have been integrating them together. This put some steep time constraints to get the project done.

Appendix A – Team Members

Table 2.0 below shows how the individual hardware component development work was assigned to each team member.

Component	Assignee
Keypad -12 button	Cody
16x2 Character LCD	Team
PIR Motion Sensor	Farnoud
Speaker	Abubakr
LED	Team
Integration	Team

Table 2.0- Team and Assignments

Appendix B – Parts List

Table 3.0 below shows the hardware used for the MSAS.

	Component	Details
1	Microcontroller	MSP430F5529
2	Motion sensor	PIR Sensor Module SE-10
3	Keypad	12 Button keypad
4	Speaker	0.2 Watt Speaker
5	LCD	16x2 Character LCD
6	Resistors	1x 10 ohm, 2x100 ohm, 1x 1 k ohm
7	Transistor	NPN (BC547)
8	LED	Basic red and green LEDs

Table 3.0-Hardware List

Appendix C – System Schematic

Figure 8.0 below shows the overall schematic of the system.

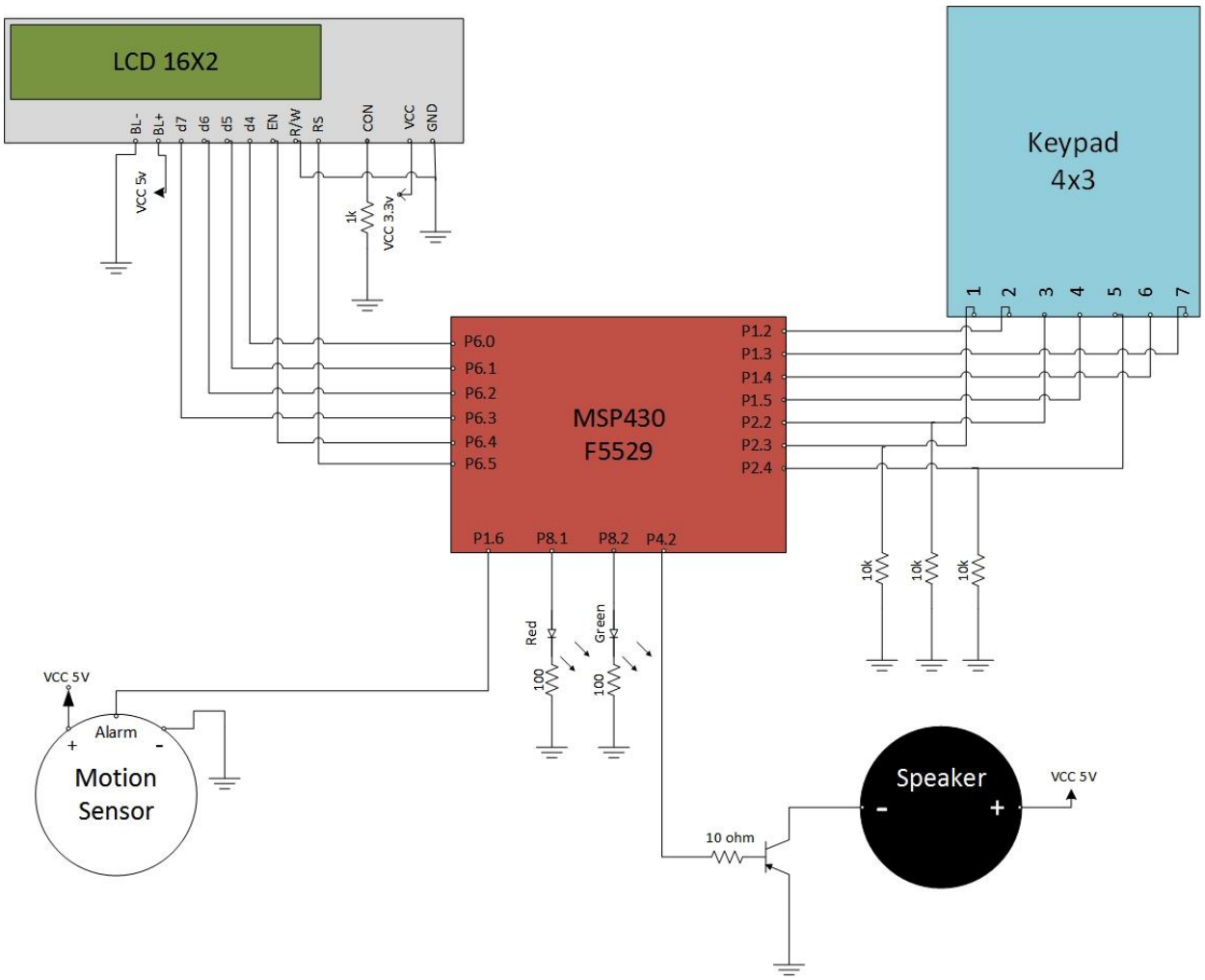

Figure 8.0-System Schematic

Works Cited

[1] E. Gurrola, "EE3376 Lab 5," [Online]. Available: http://www.ece.utep.edu/courses/web3376/Lab_5_-_LCD.html. [Accessed 29 October 2014].