Simple Home Security System

Jesus Luciano #: 016354921

Rosswell Tiongco #: 016091762

CECS 262 - Intro to Embedded Systems Programming

December 12, 2017

Table of Contents

Simple Home Security System 0

Table of Contents 1

Table of Figures 2

Introduction 3

Operation 4

Hardware 5

- Hardware Block Diagram 5
- Schematic 6
- Hardware 7
- Hardware Justification 7

Software 8

- Software Approach 8
- Software Flow Diagram 9

Conclusion 11

Table of Figures

1. Hardware Block Diagram
2. Schematic
3. State Transition Diagram
4. State Table

Introduction

The purpose of this project was to create a basic home security system utilizing everything we have learned in the semester in terms of programming an 8051 microcontroller. This project implements timers, interrupts, lcd display and input/output data manipulation.

The project traverses through its code as a moore state machine, whose outputs would be dependent on the current state. Certain inputs would change the present state to a different state and continue the cycle of changing states. The goal of the state machine was to sync and join individual components and subsystems such as a display system, and laser tripwire systems into a cohesive home security system.

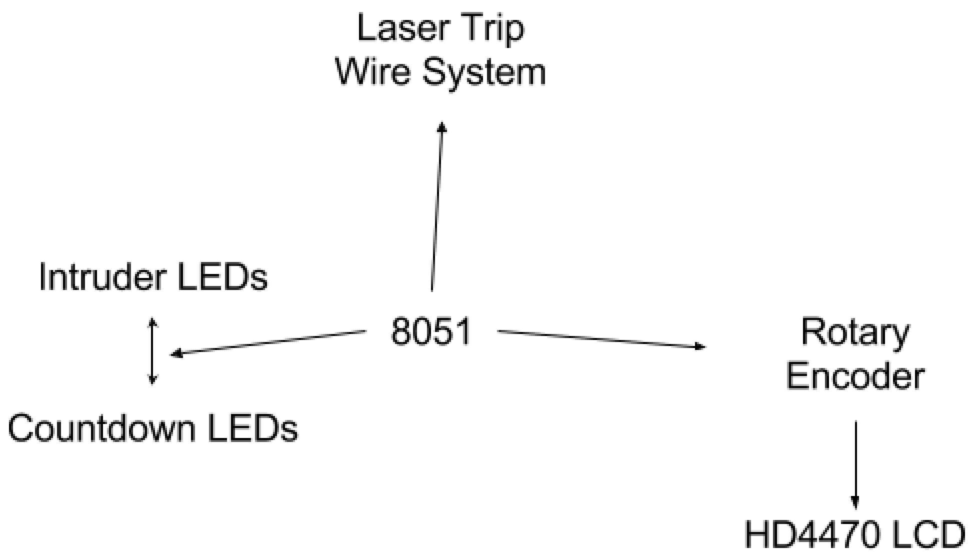
Operation

The home security system starts in its disarmed state and waits for the user to spin a rotary encoder to a desired countdown time, in seconds, which would update on the LCD screen. After being satisfied with the time, the user would then arm the security system. The LCD screen would then begin counting down the time set by the user. While the timer remained above 7 seconds remaining, a green LED would remain on. Between 6 and 4 seconds, the green LED would turn off and a yellow LED would turn on. Anytime below 3 seconds, the yellow LED would turn off and a red LED would turn off. Once the timer reached 0 seconds, a laser would turn on and hit a photoresistor. This would signify that the system is online and would wait until the path between the laser and photoresistor was interrupted by something.

Once the laser tripwire is “tripped”, the countdown timer would begin again, The user would have that many seconds, specified by the user when arming the system, to disarm the security system with a switch. If the user did not, the second line on the LCD would display “Intruder!”, 2 LEDs on the 8051 trainer board would alternate flashing at half second intervals and an alarm would sound. This would occur until the system was disarmed. If the user disarmed the system, the user would be able to adjust the time on the screen and the 2nd row of the LCD would display “Disarmed”.

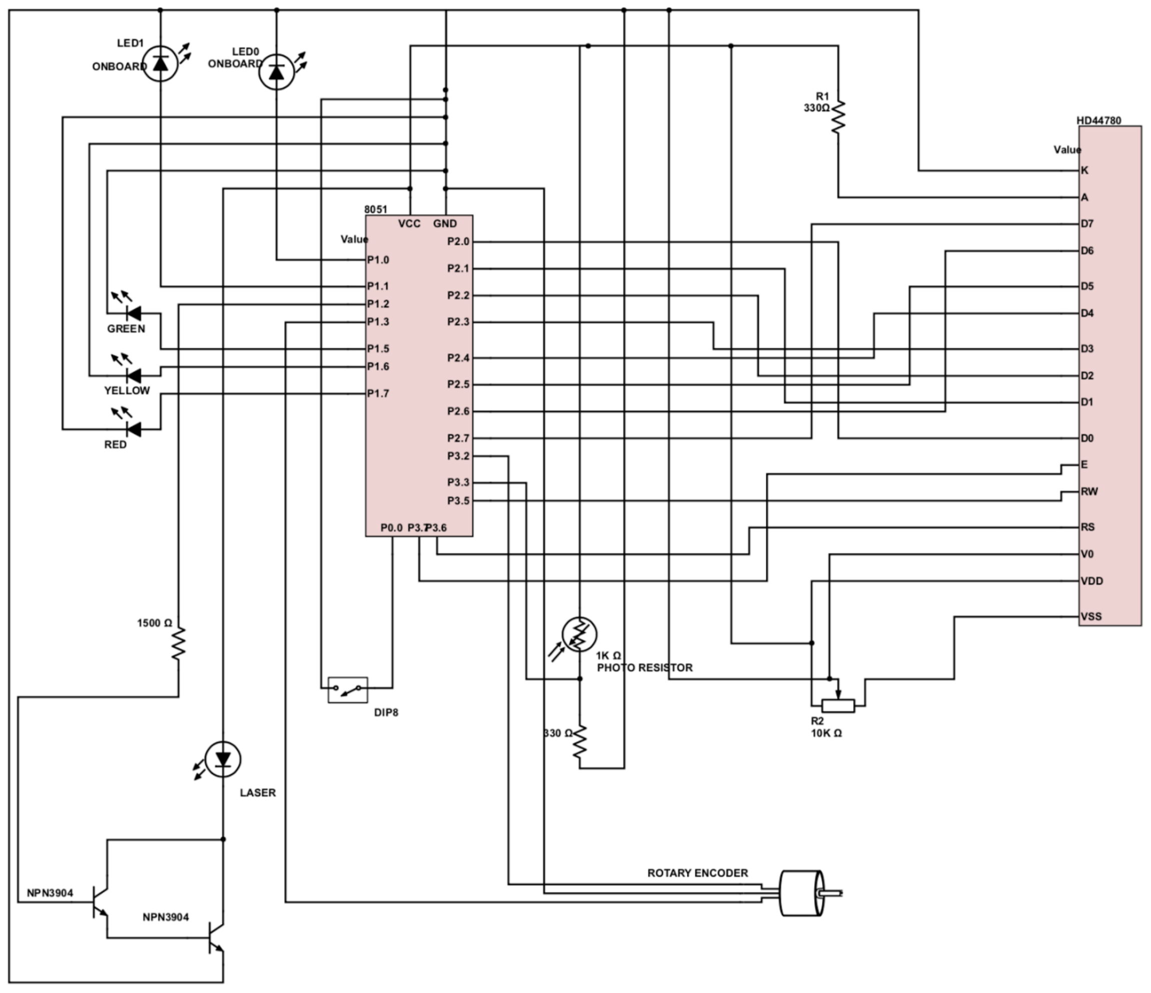
Hardware

Hardware Block Diagram



Hardware Block Diagram

Schematic



Schematic

Hardware

1. 8051 Microcontroller and its trainer board
    a. 2 x LED’s
    b. 1x8 dip switch
2. 5V Laser Diode
3. Photoresistor
4. 5v Red, Yellow and Green LEDs
5. HD44780 LCD
6. 2 x 2N3904 Transistor
7. 10k Ω Potentiometer
8. 330 Ω Resistor
9. 1500 Ω Resistor
10. Rotary encoder
11. Speaker (Piezo)

Hardware Justification

The 8051 is used to interface and establish communication between all the individual components to act cohesively. The laser trip wire system is composed of a laser diode and photoresistor. The external LEDs are used to indicate how much time the user has left to leave or disarm the system. 2 transistors were used to create a Darlington transistor. The resistors are used as voltage dividers for the transistor and laser system. In order to display information, the HD44780 LCD is used. In order to control the contrast on the LCD screen, a potentiometer is used to allow more or less current to the backlight. The rotary encoder is used to produce a continuous signal that is not directionally limited such as how a potentiometer has a limited swiping range. Finally, a piezo speaker is used to produce an audible signal that an intruder has been detected.

Software

Software Approach

The main approach followed was a top down level, wherein we designed states that would call multiple individual functions and decide what the next state should be based on an input. The states are enumerated, or declared outside of the main function. The port pin and variable declarations are also done outside of the main function. In the main function, the timer and external interrupts are enabled. Next, the superloop constantly checks for the state and carries out the designated subroutines, driving the rest of the program. When entering a new state flags are set or cleared upon entering and exiting states.

At the start of the program, the current state is initialized as the disarmed state. In the disarmed state, all components are turned off except for the lcd such as the LEDs and laser. In this state, the LCD displays the status “DISARMED”. The next state is determined by the encoder or the arming button.

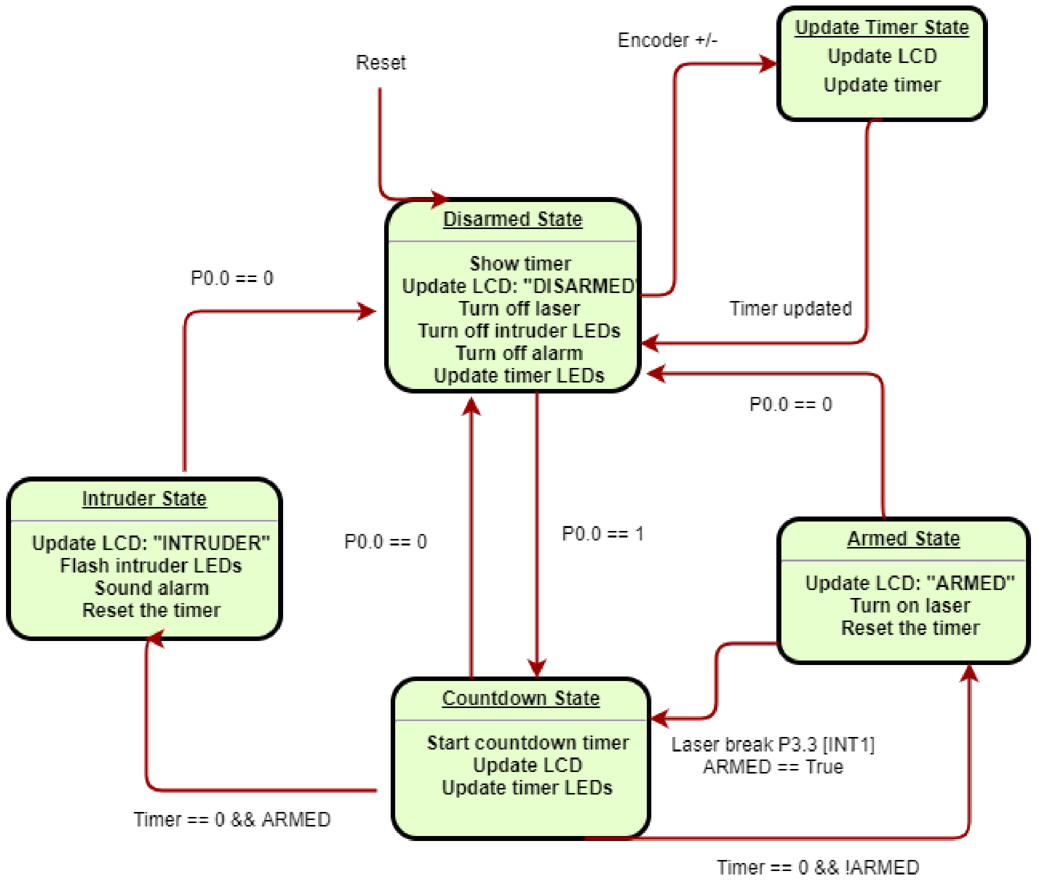
The encoder function is called whenever interrupt 0 is triggered through P3.2. When the program is interrupted, if the other encoder signal is high, the directionality is set to positive, which increments a variable. If the other encoder signal is low at the time of interrupt, the directionality is set to negative, essentially decrementing the variable countDownNum. If the encoder is incremented or decremented, we are taken to the update timer state where the lcd will output a new time based on changes made by the encoder. If the user escapes the 99 or 7 second bounds, the time is automatically hard locked to the maximum or minimum amount, respectively. Finally, the write flag is set to high, indicating that the LCD should be rewritten when the next state is set to the disarmed state.

Once the user arms the security system, we are taken to the countdown state where the LCD will countdown from the time set by the user. The delay is dictated by a 50ms timer interrupt that is iterated through 20 times, creating a 1 second delay between each count down. After every iteration of the loop, the LCD writes a value and decrements it. Based on the current value displayed to the LCD, a set of LEDs will turn on, indicating how much time is left until the system is fully armed. As long as there are more than 7 seconds left, only the green light is triggered. When there are 4 -7 seconds left, the yellow light is activated. Finally, only the red LED is turned on for the final 3 seconds. It is important to note that we programmed the LEDs to activate through an active low signal.

If the timer runs out and the system is not currently armed, we are taken to the armed state. In the armed state, we display “ARMED” onto the LCD and turn the laser on by sending a signal to the transistor. Finally, the timer and armed flags are set to high.

If the timer runs out when already in the armed state while the laser has been tripped, we are taken to the intruder state. In order to trip the laser, the photoresistor must stop receiving the laser. When this happens, an interrupt changes a laser_break flag. In the intruder state, the LCD will display “INTRUDER” and flash the 8051’s onboard LEDs and sound an alarm to further indicate the presence of an intruder. The next state logic says to remain in the armed state until the user disarms the system, which then takes the program to the disarmed state.

Software Flow Diagram



*State diagram provided by instructor, John Yu

State Table

State/ Input	Disarmed	Update	Armed	Countdown	Intruder
ARM == 0	-	-	Disarmed	Disarmed	Disarmed
ARM == 1	Countdown	-	-	-	-
Encoder	Update	-	-	-	-
ARMED && Laser_Break	-	-	Countdown	-	-
!ARMED && Timer == 0	-	-	-	Armed	-
ARMED && Timer == 0	-	-	-	Intruder	-
Timer Updated	-	Disarmed	-	-	-

State Table

Conclusion

The project focusing on designing and implementing a home security system was successfully completed. Furthermore, the project cemented the concepts of state machines, voltage dividers, and the difference between signal and power voltages. Within the code, implementing the state machine turned out very well. Finally, as the second partner project, version control was used once more and proved to be very useful in tracking changes.

The project taught lessons through failure. The main culprits for us initially failing were producing the circuits. For example, we learned that the microcontroller cannot provide a large enough current through its pins. Therefore we drove components from a powerline and toggled the component on and off through a network of transistors, using a Darlington transistor circuit.