CECS 346 Fall 2018 Project 3

A Simple Smart Home

By

Jason Chang & Justin Maeder

December 14th, 2018

This project utilizes a stepper motor and an obstacle avoidance sensor to simulate a sensor triggered garage door. The stepper motor rotates accordingly by the sensor which is triggered by objects approaching its detection range.

Introduction:

This project involves interlacing a stepper motor and an obstacle avoidance sensor to develop a working garage simulation. The goal is to use the sensor to detect an income object and then utilize the trigger flag on the sensor to rotate the motor thus simulating the action of opening a garage door. Furthermore, the push button on the board is used to simulate the doorbell as well as the two LED's which simulate the three lights in different rooms. The sensor acknowledgment is executed when the sensor's flag is set when an object is in the detection range. The sensor utilizes knowledge from lab five which changes the flag of the sensor depending on when an object is approaching or leaving and stays in its position when an object is in range and the same way when an object is not in range. By combining all of these together, we are able to create a simple smart home which includes a garage door with a doorbell that is able to open the garage door for approaching vehicles or objects via an obstacle avoidance sensor.

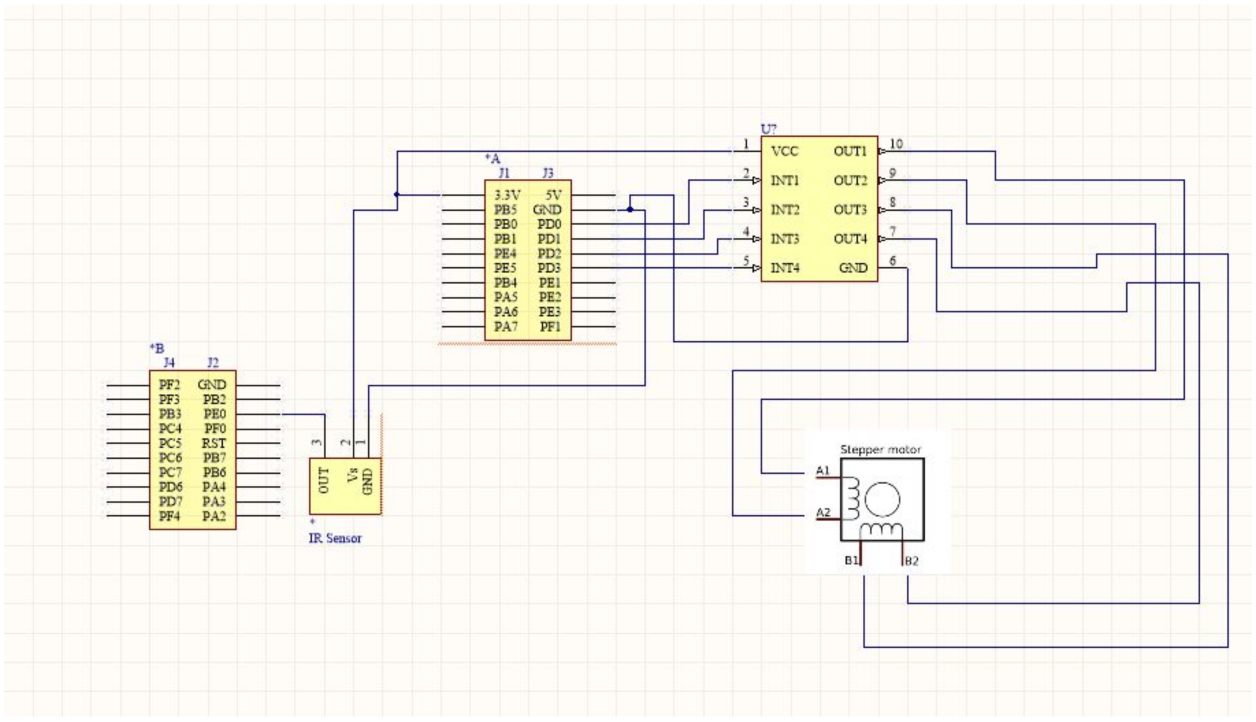
Operation:

This project utilizes a stepper motor connected to a driver that is connected to the ARMs Cortex Port D pin 0 to 3 alongside with a sensor connected to the board in order to trigger the interrupt flag when an interrupt event occurs. Upon the sensor detecting an oncoming vehicle, a hardware interrupt is triggered and a signal is sent to the board. This signal causes the stepper motor to turn counterclockwise or open the garage door and the stepper holds at the position until further detection is taken. Once the object is not within detecting range, another signal is sent to the board which triggers the motor to turn the stepper motor back to its original position. The avoidance sensor then utilizes the software from lab five which implemented the SysTick clock as well as the interrupt service routine to ensure correct detection of approaching the object and synchronize it with the turning of the garage door. The interaction between the object avoidance sensor and the stepper motor uses the built-in push buttons on the ARM board, which represents doorbells to trigger the same sequence of executions as the detecting sensor. Whenever a button is pressed, an interrupt is triggered without having an object approaching or leaving the sensor's range, and the stepper motor will turn accordingly to whichever button is pressed. The onboard LEDs, the different colors represents the current state of the sensor or push button. When no object is detected or button 1 is not pushed, the LED light will remain green. While if an object is approaching, the LED will switch from green to blinking red before switching to blue. If an object is leaving the sensor, the light switches from blue to blinking red and then staying at green. The garage makes its turns while the light is blinking red. Once the blinking cycle is over, the stepper motor stops turning and remains at its final position until next button press or detection. The blinking red signals that the sensor has detected or undetected something and the garage door is in motion. All possible combinations are demonstrated in our youtube video: https://www.youtube.com/watch?v=FB8hBDJBF8k

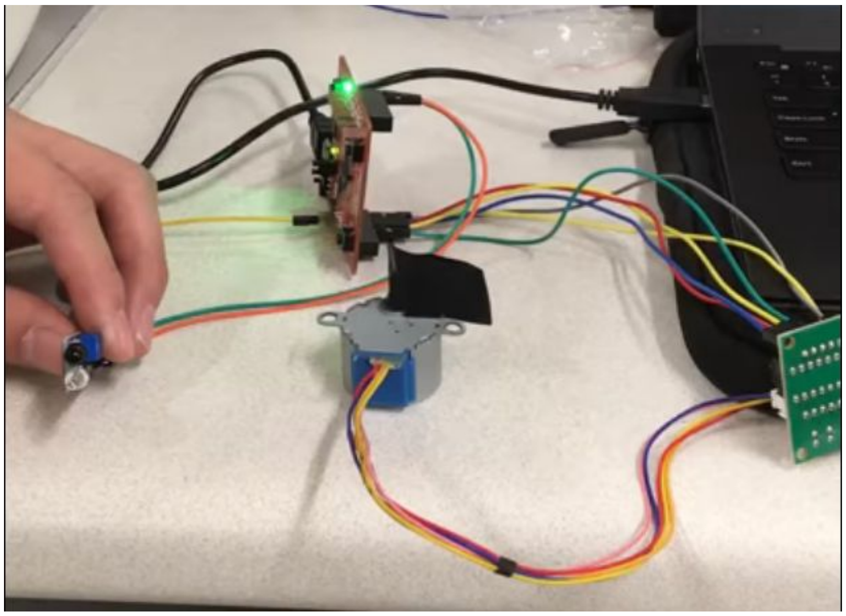
Theories:

There are two essential concepts that are used in this project which are allowing the obstacle avoidance sensor and stepper motor to work together to simulate a sensor detection triggered garage door. The garage door is designed with an ARMs Cortex board programmed with C code and wired together with an obstacle avoidance sensor, and a stepper motor connected to its driver. The onboard LEDs and push buttons represent the doorbell and lights within the rooms of the house. The project allows us to demonstrate an understanding and applies the concept of SysTick and Finite State Machine. The obstacle avoidance sensor is used for the garage to detect approaching or departing vehicles to close or open it accordingly. The default state of the LED is green which represents no detection. When an object is detected, the sensor flashes red to represent the garage door turning in motion and turns blue to stop the motion and remains blue as long as the object is within range. During the duration of red light flashing, the stepper motor makes its turns to represent the garage door opening and the turning halts as soon as the LED gets to blue. When an object leaves detection, the LED switches from blue to flashing red than to green. Once again, the flashing red represents the garage door turning while green represents that nothing is in the detection range. The onboard push buttons activate the same sequence of events as the obstacle avoidance sensor respectively. When button one is pushed, the stepper motor and LEDs response in the same series of events the same as if an object was detected by the sensor. When button 2 is pressed, the stepper motor and LEDs act as if an object is leaving the sensor detection range. In addition to all the normal functions, the sensor and buttons are required to function together and in the same manner. When an object is already detected by the sensor and the stepper motor already makes its turns, if button 1 is pressed, the LED must remain blue and the stepper motor can not turn. When an object has left the sensor and button 2 is pressed, the lights should not flash and stepper motor should not turn. The above helps ensure that the buttons are functioning as the sensor and in line with it as well.

Schematic:



Physical Design:



Software Design:

For our software design, we implemented an IR sensor and stepper motor to our TM4C board. We utilized Port E and Port F in order to set up the hardware connections. In order to trigger an interrupt, we needed to 1. arm the device, 2. NVIC enable, 3. Global Enable, 4. Set interrupts trigger priority higher than current global level defined in BASEPRI, and lastly 5. Have a hardware event trigger. Port E Pin 0 was used to control both-edge interrupt signal from the IR sensor. Inside ISR for port E, the control flags are set to be the current LEDs color from PE0. When an object enters the 15cm range of the IR sensor, the red light will flash while the motor turns 180 degrees clockwise then the light will turn blue. When the object leaves the 15cm range of the IR sensor, the red light will flash while the motor turns 180 degrees counterclockwise then the light will turn green. Port F is a falling edge interrupt that has inputs of sw1 and sw2. If sw1 or sw2 is pressed while the blue or green LED is on respectively, the LED will not change color. If sw1 is pressed while the green LED is on, the motor will turn clockwise 180 degrees while red led flashes, then the LED will turn blue. If sw2 is pressed while blue LED is on, the motor will turn counterclockwise 180 degrees and the red LED will flash which would result in the green LED turning on after the rotation. In order to rotate the stepper motor, we used an 8ms delay in between each step. Each step for the stepper motor is 1.8 degrees. In order to turn 180 degrees the turning function will run through 100 iterations. The stepper rotate functions are implemented from stepper.h file. We implemented a delay function of 100ms (a software debounce) in the Port E ISR for the input from the IR sensor. The flash function is implemented using a systick interrupt. Our Systick handler increments the “Counts” variable every 500ms. We used modulus to control the flash function which will turn on the red LED when the Counts variable is even, and turns off the LED when the count variable is odd.

<u>Conclusion:</u>

In conclusion, many problems came up along the way. The first major issue that we ran into was reading the correct signal sent from hardware back to the computer when dealing with the IR sensor. At first, we were unable to separate the both-edge signal from the IR sensor to PE0. While debugging the both-edge trigger interrupt, we thought reading the input would result in just a one-bit input whenever an object approaches or leaves the 15cm range. However, our instructor made us realize that separating the reading signal is possible by reading the Port E data register. When an obstacle approaches the range of the IR sensor, the data register outputs a falling-edge from a logic “1” to a “0”. Likewise, when an obstacle exits the range of the IR sensor, the data register outputs a rising-edge from a logic “0” to a “1”. Our second major issue occured when implementing the different combinations with sw1, sw2, and the IR sensor due to the fact that we had to implement them to work together simultaneously (specifically when sw1 is pressed while an object is in front of the sensor). The problem that occurred repeatedly was that the LED would flash when sw1 was pressed (LED was blue) and an object came entered the IR sensor range; however, the LED and the stepper motor was not meant to move with this combination. In order to fix this, we used Keil’s onboard debugger while putting breaks in our code and went line by line. After no luck, we discovered that the IR sensor we were given was producing false output signals; therefore, we went through 4 different IR sensors until we found one that did not give a false signal. The last major issue we ran into was the delay to rotate the stepper motor. The stepper motor requires its delay to be within a certain parameter to make an accurate and precise rotation according to the signal sent to it. Our problem was that the stepper motor kept vibrating in place. After decreasing the time allotted in the delay function, we were able to make our stepper motor turn 180 degrees clockwise and counterclockwise accurately. Our last small issue occurred when our stepper motor would only rotate counterclockwise and not clockwise. Again, after switching our stepper motor for a new motor, (stepper motor driver was okay) both turns worked properly. Lastly, in the duration of this project we encountered both software and hardware issues due to the IR sensor the stepper motor.