# **Project Name: Visitor Counter**

The Visitor Counter project counts the number of people currently in the room by using two ultrasonic proximity sensors that detect if a person enters the room or if a person leaves the room. This is used in many real world scenarios such as counting the amount of fish collected at a fishery, protecting buildings, counting cars in parking lots, tracking customers in places of business, and tracking the amount of people at a party. This project helped build the knowledge of how proximity sensors work and their use in the real world. Our group always had an interest on using sensors and its applications, but we never had a chance to work on it until this project.

**System Requirements**

* A box to make the counter portable for any location
* Two ultrasonic proximity sensors that detect entering and leaving
* The counter displays “total: #” on the 16 x 2 LCD screen to show the amount of people in room
* From proposal, a buzzer was going to be used, but decided to use LEDs instead to light up when being detected

## System Specifications

* The counter display increases from 0 to 1000 before resetting, since it is rare to have that many people in room and so there will be no negative total
* The arduino is powered externally with a battery pack with four AAA batteries
* The counter measures only one person at a time within half a second after person fully goes through or out for accuracy
* The HC-SR04 proximity sensors could measure distances from 0 to 40 cm, but limit set at 15-30 cm to catch when people enter or leave

## System Architecture

Figure 1 shows the high level black box diagram of the design. This high level black box diagram is slightly modified from that of the initial proposal. This shows the Arduino connected to two proximity sensors and the LCD, but instead of a buzzer, it is used with and LED. The LED was used instead of the buzzer because we wanted it to be more visual to show when people enter or leave instead of sound. This also makes the final product look more pleasing to the eye with the lights from the LED.



*Figure 1: Visitor Counter high-level block diagram*

## 

## System Design

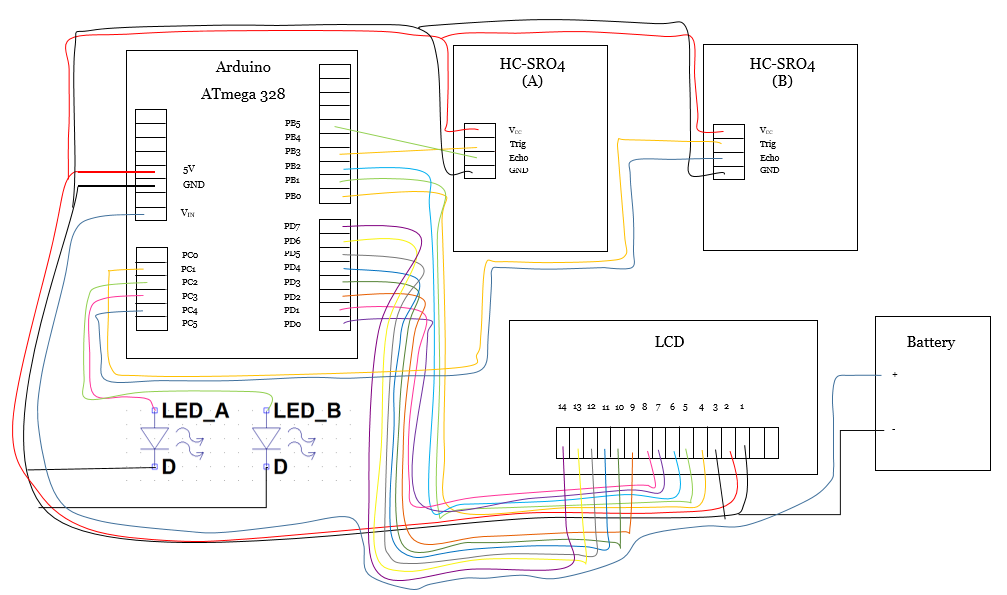
This section describes the hardware and software design of the Visitor Counter.

### 

### Hardware

* 1 Arduino Uno
* 2 2.2kOhm resistors
* 2 Pink LED high brightness
* 2 HC-SR04 Ultrasonic Sensors
* 1 LCD
* 1 Battery Pack
* 4 AAA Batteries

The schematic below (Figure 2) shows the battery connected to the Arduino’s VIN to supply power to the board. The two proximity sensors are connected to power and the corresponding trigger and echo pins are connected to various ports on the board which can be seen in Table 2 and the LCD’s pinout can be seen in Table 1 (exact same layout from Project 1). Last, the wiring table for the LEDs and battery can be seen in Table 3.



*Figure 2: Schematic of the Visitor Counter*

Table 1: Arduino Wiring for LCD

|  |  |  |  |
| --- | --- | --- | --- |
| LCD Pin | LCD Name | Arduino Pin | Port Name |
| 1 | VSS | Power: GND | GND |
| 2 | VDD | Power: 5V | VCC |
| 3 | VL | Power: GND | GND |
| 4 | RS | Digital: 8 | PB0 |
| 5 | R/W | Digital: 9 | PB1 |
| 6 | E | Digital: 10 | PB2 |
| 7 | D0 | Digital: 0 | PD0 |
| 8 | D1 | Digital: 1 | PD1 |
| 9 | D2 | Digital: 2 | PD2 |
| 10 | D3 | Digital: 3 | PD3 |
| 11 | D4 | Digital: 4 | PD4 |
| 12 | D5 | Digital: 5 | PD5 |
| 13 | D6 | Digital: 6 | PD6 |
| 14 | D7 | Digital: 7 | PD7 |

Table 2: Arduino Wiring for HC-SR04 Ultrasonic Sensors

(note that there are 2 sensors labeled A and B)

|  |  |  |
| --- | --- | --- |
| HC-SR04  Pin Name | Arduino  Pin | Port  Name |
| VCC A | Power: 5 V | VCC |
| TRIG A | Digital: 11 | PB3 |
| ECHO A | Digital: 13 | PB5 |
| GND A | Power: GND | GND |
| VCC B | Power: 5 V | VCC |
| TRIG B | Analog: A1 | PC1 |
| ECHO B | Analog: A4 | PC4 |

Table 3: Arduino Wiring for LEDs and Battery

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Component  Pin Name | Arduino  Pin | Port  Name |
| LED A | Anode | Analog: A3 | PC3 |
| LED A | Cathode | Power: GND | GND |
| LED B | Anode | Analog: A2 | PC2 |
| LED B | Cathode | Power : GND | GND |
| Battery | Positive | Power: VIN | VIN |
| Battery | Negative | Power: GND | GND |



*Figure 3: Picture of Final Design (wires are hidden inside, refer to schematic for layout and pin tables for wiring)*

Figure 3 above shows the final design with the LCD in the middle, displays the number of people in the room, with sensor A on the bottom left, sensor B on the bottom right, to detect the people, and the two LEDs in between them indicating if the sensor got triggered by a person. Not shown is the back of the box which contains a switch for the battery that supplies power to the arduino. The box is made of simple cardboard and one of the finest materials found in SLO.

**Software**

The software used to develop the Visitor Counter was created with C using Atmel Studio and AVR. To get the sensors working, the MCU must send a high to low signal to the sensor and when then produce an echo back. The time it takes for the echo to return to the sensor is then used to find the distance the object is away from the sensor. In the software, send high and low to the TRIG pin and then use polling to count when the ECHO pin is high, but stop when it goes low. This is done to both sensors and then use conditional logic to determine if the person is entering or leaving by finding out which sensor is activated first. Then display the count on the LCD. An example of the software to develop the sensor is shown below in Figure 4 with visuals to show the trigger pulse, the transmitted pulse from the sensor, and the time it takes for the pulse to get back to be used to calculate distance and then overall detection sensing.

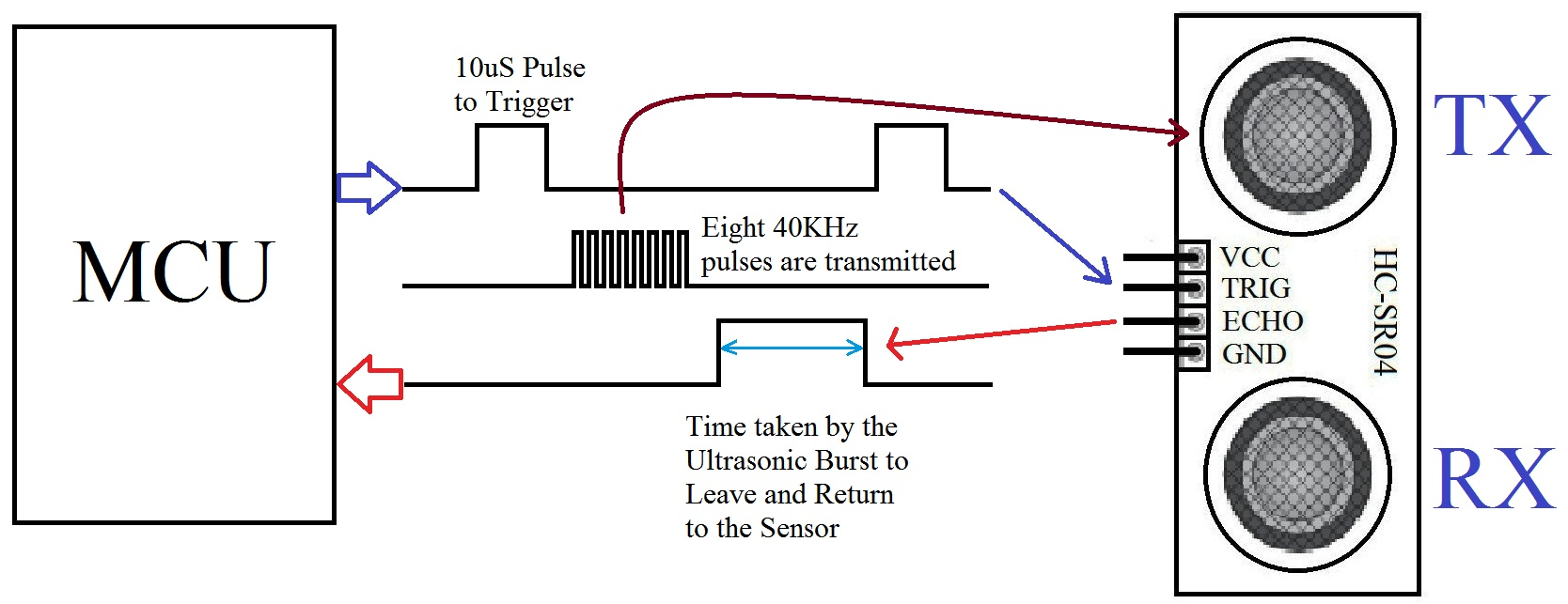
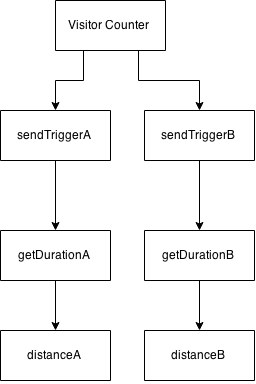


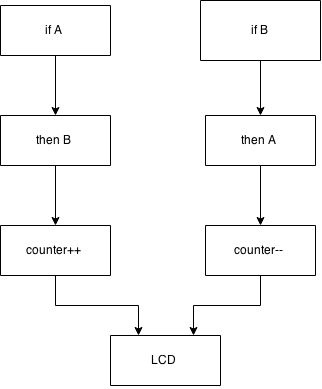
Figure 4: Example of how HC-SR04 works

In Figure 5 below, function sendTriggerA and sendTriggerB send the pulse from the MCU to the sensors. getDurationA and getDurationB will use polling and determine how long it takes to get pulse back to echo. Lastly, distanceA and distanceB is the conversion from time (microseconds) to distance (centimeters).



*Figure 5: Simple Code map of calculating distance from sensors*

In Figure 6, the conditional block is what determines if a person is entering or leaving. This project only correctly works if a person completely goes in and completely goes out with no other option. From A to B, a person will be entering the room making the count go up. From B to A, a person will be leaving the room making the count go down.



*Figure 6 : Conditional Block for entering and leaving*

## Test Results

This section describes how each requirement or specification for the Visitor Counter was met or not met.

Table 4: Test Results

|  |  |  |
| --- | --- | --- |
| Requirement/Specification | Test Used | Results, Passed? |
| Accurate Distance Measurement from Sensors | Used a ruler and a book to see if the sensor and LCD output the right distance compared to the ruler | Yes, this is actually very accurate from 0-40 cm. |
| LEDS light up when the sensor detects a person | Distance range was 15-30 cm and checked if the LED would turn on when putting hand in that range | Yes the first sensor would pick up the hand instantly and light up the LED, but the second sensor would detect about 0-.5 seconds after due to software delays, but sometimes it would instantly light up |
| Be able to detect two people at once | First had two people walk at same time shoulder to shoulder through the sensors and then tried with half a second after and then a second after. | Two people at same time failed as well as walking half a second apart. This can be improved with more time and software improvements, but the interval of a second after is sufficient and counts correctly. There is about 0.5 seconds delay needed to get accurate reading |
| Make the project portable so that it can be used anywhere such as a door | Used a battery pack with 4 AAA batteries and tested to see if the arduino still got powered by it. Then designed a box to place the components in and checked to see if the detection worked at various different locations | The battery pack worked perfectly and the box design also worked and since it is powered externally, it can be placed anywhere desired and can be turned on and off with the battery pack. |

## 

## Personnel

Anthony Dinh is a third year Computer Engineering student and Michael Jurs is a third year Electrical Engineering student. Anthony’s main skill and expertise is coding in java and in C and contributes to the group towards the software development part of the project. Anthony and Michael both thought of the concept of the visitor counter. The developing the hardware connections, debugging the circuit, testing the device, and writing the low-level driver, user interface, and high-level programs were completed by both students. The powerpoint presentation and report were also completed by both Anthony and Michael.

## 

## Conclusion

### Anthony Dinh

This project idea was started after watching several videos online on arduino projects because proximity sensors seemed to have really cool properties and had a variety of applications it could be used for. I specifically thought that the sensor was unique and I wanted to learn more about how it worked by making the Visitor Counter. Initially, our group planned on having a counter that could be attached to a door to detect one to two people at a time and count accurately with no delay in between at all. A buzzer would tell the user when a person entered or left at different pitches. However, coming from that initial idea, we decided to have the project be structured as a box so that it could be more portable and can be placed almost anywhere. Therefore, it means it could not only be placed at a door, but anywhere with a flat surface. The one to two people counting didn’t work as plan and found out that the Visitor Counter could only count about one person per half a second. The idea of having a buzzer was left out since the sensor itself took more time to setup and code then we thought and ended up using LEDS to visually show that the detection was working. However, LEDS seemed to be pleasing to the eye and easier to notice. As mentioned earlier, the challenges that arose from this project was the coding of the two sensors. Initially, one sensor was easily coded by using two pin change interrupts and 2 overflow interrupts, but when adding the second sensor, it was too slow to detect a person. In the end, polling was a better option for the software and it was faster. Other challenges included trying to get almost no time delay (shortest amount of time) for a person to enter or leave through the sensors, but this was not possible due to software delays written in the code. If there was more time to work on project, we would include the buzzer in conjunction with the LEDS to have a sound be played when people entered or left the room, rewrite the code so that less delays are present and have the counter be able to detect multiple people at once. Also, to use better materials to make the box that the counter is made out of look more pleasing to the eye. More time can also allow to fix random bugs that occur such as double counting and even add more features in such as storing the number of people in the room at a certain time and being able to look at it later. For example, if the user of the Visitor Counter set up the box at a door at his party to determine how many people came, but wanted to see the distribution of people at certain times, the user could press a button to see on the LCD how many people there were at the party at 12:00, 1:00, etc. Overall, this project was a fun experience and helped combined ideas from earlier projects in the quarter such as LCD, interrupts, timers, and delays.

### Michael Jurs

The Visitor Counter project counts the number of people currently in the room by using two ultrasonic proximity sensors to detect whether a person is entering or leaving a room. This project has several important applications including counting the amount of fish collected at a fishery, counting individuals entering buildings for security, counting cars in parking lots, counting customers in places of business, and counting the amount of people at a party. This project helped build the knowledge of how proximity sensors work and their use in the real world. Our group had an interest to use sensors and explore its applications, but we did not have a chance to work on it until this project. When I finish my undergraduate coursework I would like to go to graduate school and study Neural Engineering or find a job working for a medical device company and going to graduate school a few years later. Currently, embedded systems are used in devices such as glucose testers and magnetic resonance imaging (MRI). The group spent a lot of time trying to figure out how to incorporate two sensors to work simultaneously. Eventually, we realized with the help of our instructor, that it must be the way in which we were calling the function. The code would check for sensor A more frequently than it would for sensor B, thus starving B and making it inaccurate or even fail. The fact that this project was created by the students themselves provided the mindset that it was not“boring” classwork, but rather a project that would result in fulfillment for the students. Adding the LEDs and the LCD made the project fun because it allowed us to test our project visually, reducing the time we needed to read many lines of code. If we had more time, the project could be improved to be some sort of alarm for the visually impaired. If a buzzer was added to the circuit, then if someone were about to stumble into a wall, a sound would alarm the person that they will hurt himself or herself. Another application could be to help a professor count the number of animals they are researching. The only remaining task is for the counter to be able to count multiple people simultaneously. Ultimately, this project did a great job synthesizing the information learned throughout the quarter.

## 

## References

## ATmega 328 Datasheet (on PolyLearn)

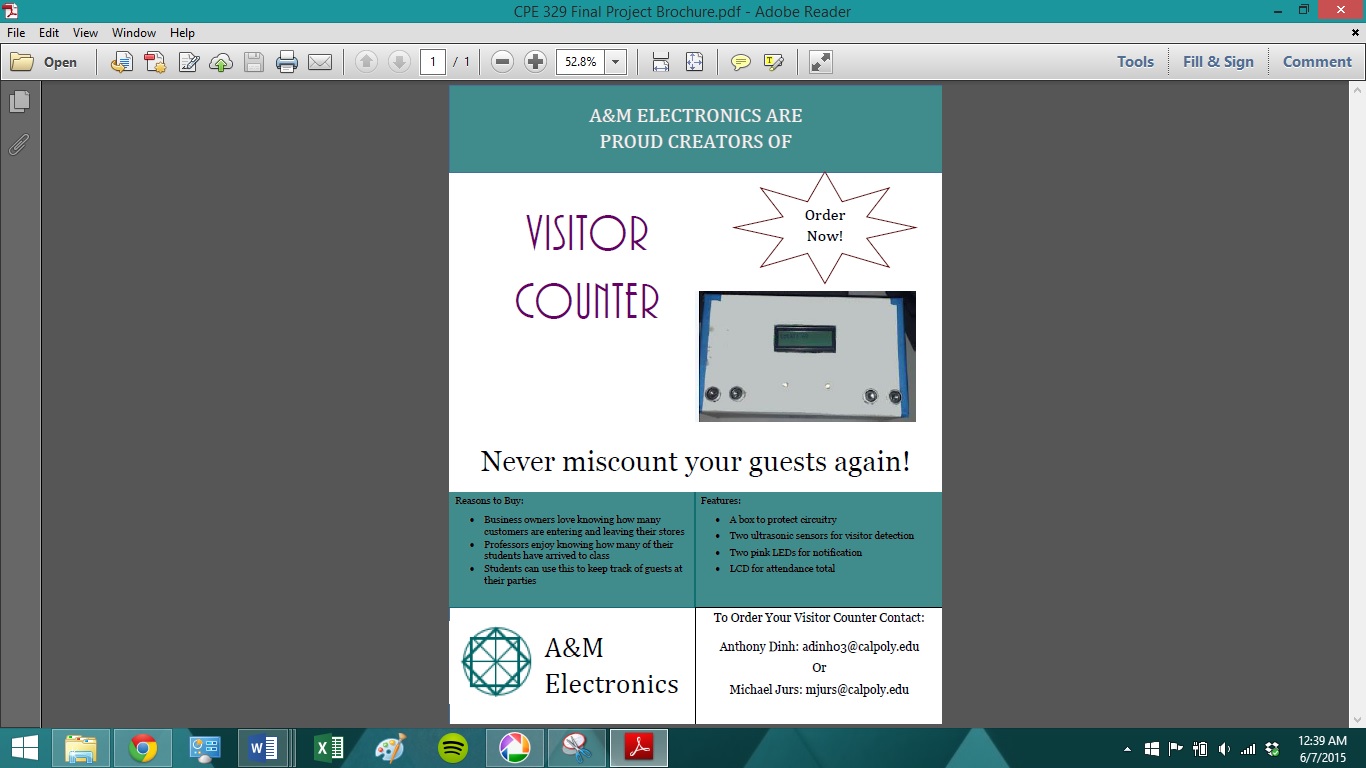
LCD Datasheet (on PolyLearn)

HC-SR04 Datasheet (http://www.micropik.com/PDF/HCSR04.pdf)

## 

## 

## Appendix A – Technical or Marketing Document



## Appendix B – Source Code

The file will also be uploaded for better view

/\* file Sensor.c  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
Visitor Counter: counts the amount of people entering or leaving a room  
  
IDE: Atmel Studio 6.2  
  
Target: ATmega328P 8b microcontroller on Arduino Uno dev board  
  
================================================================================  
  
DEVELOPMENT TEAM  
  
Anthony Dinh  
  
Michael Jurs  
  
================================================================================  
  
CODE MAP  
  
void sendTriggerA(void); ->send trigger pulse to sensor A  
  
void sendTriggerB(void); ->send trigger pulse to sensor B  
  
unsigned long getDurationA(void); -> get the duration of echo A  
  
unsigned long getDurationB(void); -> get the duration of echo B  
  
long microsecondsToCentimeters(long); -> convert time into distance  
  
conditional block that determines who enter + leaves by looking at a certain distance  
  
void display\_count(int); -> display amount of count  
  
================================================================================  
  
TODO  
  
Add code to improve delay between people  
  
Add code to have buzzer implemented  
  
Add feature that stores about of people at a certain time period for future reference  
  
================================================================================  
  
\*/  
  
  
  
  
  
#define F\_CPU 16000000UL  
  
#include <avr/io.h>  
  
#include<avr/wdt.h>  
  
#include<avr/interrupt.h>  
  
#include<util/delay.h>  
  
#include <stdlib.h>  
  
  
  
void sendTriggerA**(**void**);**  
  
void sendTriggerB**(**void**);**  
  
unsigned long getDurationA**(**void**);**  
  
unsigned long getDurationB**(**void**);**  
  
long microsecondsToCentimeters**(**long**);**  
  
void lcd\_init**(**void**);**  
  
void lcd\_display\_distance\_for\_testing**(**long**);**  
  
void lcd\_wr\_cmd**(**char**);**  
  
void lcd\_wr\_char**(**char**);**  
  
void display\_count**(**int**);**  
  
  
  
int main**()**  
  
**{**  
  
  
  
//Initialize variables for main use  
  
long durationA**,** durationB**,** distanceA**,** distanceB**;**  
  
char lastFlag **=** 'z'**;**  
  
int count **=** 0**;**  
  
int current\_count **=** 0**;**  
  
  
  
  
  
//LCD initializing  
  
DDRB **=** 0b00000111**;** //E, RW, RS  
  
DDRD **=** 0xFF**;** //LCD Inputs  
  
  
  
//triggerB initializing  
  
DDRC **|=** **(**1 **<<** PC1**);** // output trigger b  
  
DDRC **&=** **~(**1 **<<** PC4**);** //input echo b  
  
DDRC **|=** **(**1 **<<** PC2**);** //OUTPUT LED pink  
  
  
  
//triggerA initializing  
  
DDRB **|=** **(**1 **<<** PB3**);** //trigger a  
  
DDRB **&=** **~(**1 **<<** PB5**);** //input echo a  
  
DDRC **|=** **(**1 **<<** PC3**);** //led red  
  
  
  
lcd\_init**();** //Initialize LCD to be used  
  
  
  
sei**();** //set interrupt  
  
  
  
**while** **(**1**)**  
  
**{**  
  
// this block of code sets up sensor to be used and gets the duration back  
  
sendTriggerA**();**  
  
durationA **=** getDurationA**();**  
  
sendTriggerB**();**  
  
durationB **=** getDurationB**();**  
  
  
  
//converts duration into distance  
  
distanceB **=** microsecondsToCentimeters**(**durationB**);**  
  
distanceA **=** microsecondsToCentimeters**(**durationA**);**  
  
  
  
//conditional block that detects if entering or leaving by using  
  
//a flag to determine which was the last sensor used  
  
**if** **(**distanceB **>** 15 **&&** distanceB **<** 40**)**  
  
**{**  
  
PORTC **|=** **(**1 **<<** PC2**);**  
  
**if** **(**lastFlag **==** 'a'**)**  
  
**{**  
  
count**++;**  
  
lastFlag **=** 'z'**;**  
  
\_delay\_ms**(**100**);**  
  
**}** **else**  
  
**{**  
  
lastFlag **=** 'b'**;**  
  
**}**  
  
  
  
**}** **else**  
  
**{**  
  
PORTC **&=** **~(**1 **<<** PC2**);**  
  
**}**  
  
  
  
**if** **(**distanceA **>** 15 **&&** distanceA **<** 40**)**  
  
**{**  
  
PORTC **|=** **(**1 **<<** PC3**);**  
  
**if** **(**lastFlag **==** 'b'**)**  
  
**{**  
  
count**--;**  
  
lastFlag **=** 'z'**;**  
  
\_delay\_ms**(**100**);**  
  
**}** **else**  
  
**{**  
  
lastFlag **=** 'a'**;**  
  
**}**  
  
  
  
**}** **else**  
  
**{**  
  
PORTC **&=** **~(**1 **<<** PC3**);**  
  
**}**  
  
  
  
// if over 1000 people in room just restart counter  
  
//not realistically possible  
  
//and not allowed to have negative count  
  
**if** **(**count **==** 1000 **||** count **<** 0**)**  
  
**{**  
  
count **=** 0**;**  
  
**}**  
  
  
  
//displays count to LCD  
  
**if** **(**count **!=** current\_count**)**  
  
**{**  
  
current\_count **=** count**;**  
  
lcd\_wr\_cmd**(**0x01**);**  
  
display\_count**(**count**);**  
  
  
  
**}**  
  
  
  
\_delay\_ms**(**200**);**  
  
  
  
  
  
**}**  
  
**}**  
  
  
  
//sending pulse from MCU to sensor B  
  
void sendTriggerB**()**  
  
**{**  
  
  
  
PORTC **&=** **~(**1 **<<** PC1**);**  
  
\_delay\_us**(**2**);**  
  
PORTC **|=** **(**1 **<<** PC1**);**  
  
\_delay\_us**(**5**);**  
  
PORTC **&=** **~(**1 **<<** PC1**);**  
  
**}**  
  
  
  
//sending pulse from MCU to sensor A  
  
void sendTriggerA**()**  
  
**{**  
  
  
  
PORTB **&=** **~(**1 **<<** PB3**);**  
  
\_delay\_us**(**2**);**  
  
PORTB **|=** **(**1 **<<** PB3**);**  
  
\_delay\_us**(**5**);**  
  
PORTB **&=** **~(**1 **<<** PB3**);**  
  
**}**  
  
  
  
//convert time it takes to distance  
  
long microsecondsToCentimeters**(**long microseconds**)**  
  
**{**  
  
  
  
**return** microseconds **/** 29 **/** 2**;**  
  
**}**  
  
  
  
//calculates the time it takes for echo pin B to go low and returns it in microsecond  
  
unsigned long getDurationB**()**  
  
**{**  
  
unsigned long duration **=** 0**;**  
  
unsigned long loopCount **=** 0**;**  
  
unsigned long maxCount **=** 3000**;** //amount of loops desired to wait for  
  
  
  
// While the pin is not high, don't do anything and make sure the max hasnt been reached  
  
**while(!(**PINC **&** **(**1 **<<** PC4**)))**  
  
**{**  
  
**if** **(**loopCount**++** **==** maxCount**)**  
  
**{**  
  
**return** 0**;**  
  
**}**  
  
**}**  
  
  
  
// When the pin is high, increment the timer and make sure not more than max  
  
**while** **(**PINC **&** **(**1 **<<** PC4**))**  
  
**{**  
  
**if** **(**loopCount**++** **==** maxCount**)**  
  
**{**  
  
**return** 0**;**  
  
**}**  
  
duration**++;**  
  
**}**  
  
  
  
// Return the pulse time in microsecond!  
  
**return** duration **\*** 1.455**;**  
  
  
  
**}**  
  
  
  
  
  
  
  
unsigned long getDurationA**()**  
  
**{**  
  
unsigned long duration **=** 0**;**  
  
unsigned long loopCount **=** 0**;**  
  
unsigned long maxCount **=** 3000**;**  
  
  
  
// While the pin is \*not\* in the target state we make sure the timeout hasn't been reached.  
  
**while** **(!(**PINB **&** **(**1 **<<** PB5**)))**  
  
**{**  
  
**if** **(**loopCount**++** **==** maxCount**)**  
  
**{**  
  
**return** 0**;**  
  
**}**  
  
**}**  
  
  
  
// When the pin \*is\* in the target state we bump the counter while still keeping track of the timeout.  
  
**while** **(**PINB **&** **(**1 **<<** PB5**))**  
  
**{**  
  
**if** **(**loopCount**++** **==** maxCount**)**  
  
**{**  
  
**return** 0**;**  
  
**}**  
  
duration**++;**  
  
**}**  
  
  
  
// Return the pulse time in microsecond!  
  
**return** duration **\*** 1.455**;**  
  
  
  
**}**  
  
  
  
void lcd\_wr\_cmd**(**char cmd**)** // function set  
  
**{**  
  
PORTB **=** 0b00000000**;** // 0, 0, 0  
  
PORTD **=** cmd**;** //function set  
  
  
  
\_delay\_us**(**1**);** // tw delay  
  
PORTB **=** 0b00000100**;** // 1, 0 ,0 ( E is high)  
  
PORTB **=** 0b00000000**;** // 0, 0 ,0 (E go low again)  
  
  
  
**}**  
  
  
  
void lcd\_init**(**void**)**  
  
**{**  
  
\_delay\_ms**(**1**);**  
  
\_delay\_ms**(**40**);** // can be tuned down to be 20  
  
lcd\_wr\_cmd**(**0x38**);** // function set command  
  
\_delay\_us**(**80**);**  
  
lcd\_wr\_cmd**(**0x0C**);** //display set  
  
\_delay\_us**(**80**);**  
  
lcd\_wr\_cmd**(**0x06**);** //enable set  
  
\_delay\_us**(**80**);**  
  
//lcd\_wr\_custom(0, myHeart); //write to the CG RAM  
  
  
  
**}**  
  
  
  
void lcd\_wr\_char**(**char character**)**  
  
**{**  
  
PORTB **=** 0b00000000**;**  
  
PORTD **=** character**;** //display character  
  
PORTB **=** 0b00000001**;** // 0, 0, 1  
  
\_delay\_us**(**1**);** // tw delay  
  
PORTB **=** 0b00000101**;** // 1, 0 ,1  
  
PORTB **=** 0b00000001**;** // 0, 0 ,1  
  
\_delay\_ms**(**1**);**  
  
  
  
**}**  
  
//not used for final production but can be used while debugging and if wanted to see  
  
//how far sensor could reach and the distance it outputs  
  
//also can check for accuracy  
  
void lcd\_display\_distance\_for\_testing**(**long distance**)**  
  
**{**  
  
char my\_string**[**7**];**  
  
ltoa**(**distance**,** my\_string**,** 10**);**  
  
**if** **(**distance **<** 100**)**  
  
**{**  
  
**for** **(**int i **=** 0**;** i **<** 2**;** i**++)**  
  
**{**  
  
lcd\_wr\_char**(**my\_string**[**i**]);**  
  
**}**  
  
**}** **else**  
  
**{**  
  
**for** **(**int i **=** 0**;** i **<** 3**;** i**++)**  
  
**{**  
  
lcd\_wr\_char**(**my\_string**[**i**]);**  
  
**}**  
  
**}**  
  
  
  
**}**  
  
  
  
//outputs the total count to LCD  
  
void display\_count**(**int count\_input**)**  
  
**{**  
  
char count\_string**[**7**];**  
  
char unit\_message**[]** **=** **{** 't'**,** 'o'**,** 't'**,** 'a'**,** 'l'**,** ':'**,** ' ' **};**  
  
ltoa**(**count\_input**,** count\_string**,** 10**);** // function from stlib.h to covert long to char  
  
\_delay\_ms**(**100**);**  
  
  
  
//prints out "total" to LCD  
  
**for** **(**int i **=** 0**;** i **<** 7**;** i**++)**  
  
**{**  
  
lcd\_wr\_char**(**unit\_message**[**i**]);**  
  
\_delay\_ms**(**100**);**  
  
**}**  
  
  
  
//conditionals to print out number because has a funny character if digit has 2 places or 3 places  
  
**if** **(**count\_input **<** 10**)**  
  
**{**  
  
**for** **(**int i **=** 0**;** i **<** 1**;** i**++)**  
  
**{**  
  
lcd\_wr\_char**(**count\_string**[**i**]);**  
  
// \_delay\_ms(100);  
  
**}**  
  
**}** **else** **if** **(**count\_input **<** 100**)**  
  
**{**  
  
**for** **(**int i **=** 0**;** i **<** 2**;** i**++)**  
  
**{**  
  
lcd\_wr\_char**(**count\_string**[**i**]);**  
  
// \_delay\_ms(100);  
  
**}**  
  
  
  
**}** **else**  
  
**{**  
  
**for** **(**int i **=** 0**;** i **<** 3**;** i**++)**  
  
**{**  
  
lcd\_wr\_char**(**count\_string**[**i**]);**  
  
**}**  
  
**}**  
  
  
  
lcd\_wr\_char**(**' '**);**  
  
// \_delay\_ms(100);  
  
**}**