Appendix A Requirements and Verification Table

Table A.1 Project Requirements and Verification

Requirements	Verification
Power Supply: 1. Output voltage of A23 batteries should be 12 $\pm$ 1 V in order to provide steady power supply to motors. 2. 5 V voltage regulator (connected to Arduino and sensors) should output voltage of 5 $\pm$ 0.3 V with maximum current 1.5 A. 3. 3.3 V voltage regulator should supply 2.2 – 3.8 V to transceivers while current maintained within 17 $\pm$ 2 mA. 4. Output voltage of the 1K Ohm potentiometer needs to be restricted within 1.8 – 2.2 V, which provides the working current of the LED.	1. Connect the probes of a multimeter to the ends of each battery to measure the output voltage. The multimeter should read 12 $\pm$ 1 V. 2. After wiring a 5 V voltage regulator to an A23 battery, measure the output voltage with a multimeter. Expected value is 5 $\pm$ 0.5 V. Connect a 3.5 ohms resistor as load, measure the current through the load to test if the regulator can sustain a 1.5 A current. 3. Connect the 3.3 V voltage regulator to an A23 battery, use a multimeter to measure the terminal voltage of the regulator. Make sure the voltage falls in the 2.2 – 3.8 V range. Use a 200 ohms resistor as load and test if the regulator functions properly with 17 $\pm$ 2 mA current. 4. Connect the potentiometer to the SDO pin of a transceiver and adjust the resistor value. Use a multimeter to measure its output voltage when SDO is set to high. The multimeter should read 1.8 – 2.2 V.
Transceiver: 1. Transceivers should cover an effective range of 10 $\pm$ 0.5 m with no more than 5% distortion.	1. Place two transceivers 10 m apart and make one serves as receiver and the other as transmitter. Use two oscilloscopes to monitor the waveforms sent and received

2. Working at 433 MHz band, receiver bandwidth is fixed at 67 $\pm$ 5 kHz, while transmitter frequency deviates from the carrier frequency at 45 $\pm$ 3 kHz. 3. Data rate should be maintained at 1.2 $\pm$ 0.2 kbps.	by the transceivers, respectively. The received signal should differ from the origin one by less than 5%. 2. Set all transceivers and a spectrum analyzer to operate at 433 MHz. The analyzer is expected to receive the signal at frequency within $\pm$ 36 kHz after one transceiver sends a signal at 433 MHz. Then retrieve the signals received by other transceivers using the microcontroller to check if they deviate from 433 MHz by 45 $\pm$ 3 kHz. 3. Under transmitter operation, measure the interval between signals sent by a transceiver using an oscilloscope. Desired interval should be 0.71 – 1 ms.
Motor: 1. The cart should move at 1.5 $\pm$ 0.1 m/s, 1.4 $\pm$ 0.1 m/s and 1.3 $\pm$ 0.05 m/s for the three speed levels. 2. Percentage error of the actual turning angle of the front ball wheel cannot exceed 10%.	1. For each of the three levels, let the cart move for 5 s in open area. Measure the distance that the cart goes by a metric ruler and divide the result by 5. The expected results should be 1.5 $\pm$ 0.1 m, 1.4 $\pm$ 0.1 m and 1.3 $\pm$ 0.05 m, respectively. If test fails: Use a stroboscope to measure the rotation speed of the motor. The stroboscope should read 1$\pm$0.02 Hz for 1.5 m/s level, 0.93$\pm$0.02 Hz for 1.4 m/s and 0.87$\pm$0.02 Hz for 1.3 m/s. 2. Mark the start position of the cart and set the turning angle from 0 to 90° with 5° increment each time. Let the cart move for

	5 s in open area. Measure the angle between the cart's path and the direction it originally faced by a protractor. The calculated error percentage should be less than 10%.
Sensor: 1. Under 5 V DC voltages, sensor will operate at a current of 15 ± 2 mA. 2. Sensor should be able to detect objects 0.5 ± 0.05 m and 0.2 ± 0.02 m away and differentiate these two distances by triggering voltage signals of different magnitudes. 3. Working frequency of the sensor should be 40 ± 4 Hz to provide updated information of obstacles.	1. Apply 5 V DC voltages to each sensor, measure the passing current by a multimeter. The multimeter should read 15 ± 2 mA. 2. Place an object at 0.5 ± 0.05 m in front of a sensor and measure the voltage response using an oscilloscope. Repeat the same process after moving the object to 0.2 ± 0.02 m. The sensor is expected to trigger steady voltages in both cases, as long as the distance is constant. Furthermore, the former voltage should be smaller than the latter one by more than 0.2 V. These two voltages will be recorded as threshold values. 3. Set the rotation frequency of a rotor with only one tooth to 36 – 44 Hz and fix the sensor at any point that could detect the tooth. Use an oscilloscope to monitor the output voltage of the sensor. If the voltage signal is steady and constant, the sensor frequency is the same as the rotor, which means the specification is satisfied.
Microcontroller: 1. Microcontroller needs to calculate the positions of the cart and customer based on information collected from	1. Pick arbitrary positions for the cart and customer and record their coordinates. Export the positions determined by the microcontroller to a computer and check if

transceivers, and output speed and turning angle of the cart. The calculated positions should not deviate from the real positions by more than 5 cm. 2. If an obstacle is detected within 0.2 m in front or the left, the microcontroller should modify the turning angle by 15° to the right until all obstacles are cleared; if something is found in the right, increase the angle to the left by 15°. 3. Microcontroller should be able to determine which button is pressed on the keypad and correctly reflected that information on the LCD screen. 4. If the number of items entered is less than 6, the microcontroller should determine the optimal solution in 3 s; if the number is between 7 and 10, the processing time is restricted in 5 s. 5. When the distance between the cart and customer exceeds 2 m, the microcontroller will send a 'high' signal through cart's transceiver to customer's transceiver, which provides voltage to light the red LED.	they are within 5 cm of the real locations. Distances are measured based on geometric centers. 2. Place an object at a distance of less than 0.2 m from the cart in each of the three sides. Check if the microcontroller makes correct turning instructions by exporting data to a computer. If test fails: Measure the output voltage of the corresponding pins by a multimeter, which should read $\pm(0.83\pm0.05)$ V. Positive voltages indicate right-turning and negative voltages are for left-turning. 3. Press each button on the keypad and write Arduino codes to test if the microcontroller correctly decodes which button is pressed. Display the corresponding product names stored in the microcontroller on LCD. Make sure they match the names assigned to each button. If test fails: Measure the voltage of the pins connected to the LCD by a multimeter. They should be properly turned on and off according to the programming codes of Arduino, which display correct product names. 4. Test the case of n = 6 and n = 10. Trace the 'ready' signal of Arduino using an oscilloscope. Time delay should be within 3 s and 5 s, respectively.

	5. Set the distance between the cart and customer to be larger than 2 m. Measure the output voltage of the corresponding pin of Arduino and check if it is set to high.
User Interface: 1. LCD display needs be refreshed in 0.5 s after a new item is entered by keypad. And the information displayed should be the same as what the microcontroller outputs. 2. All output pins of the keypad should produce correct low/high signals in order to properly reflex which button is pressed.	1. Connect the LCD display to the microcontroller. Use an oscilloscope to measure the time delay for the LCD to output a high signal after one button is pressed on the keypad. That time interval needs to be less than 0.5 s. 2. For each button on the keypad, connect the corresponding two pins that should be set to high if that button is pressed to a multimeter. If both of them are 'on' and the remaining five are 'off', the keypad is working properly.
Alarm: 1. The red LED is expected to light when the distance between the cart and customer exceeds 2 m.	1. Set the distance between the cart and customer to be larger than 2 m. Measure the output voltage of the potentiometer that is connected to the customer transceiver. The expected value is 1.8 – 2.2 V.

Appendix B Transceiver Schematics and Wired Circuit

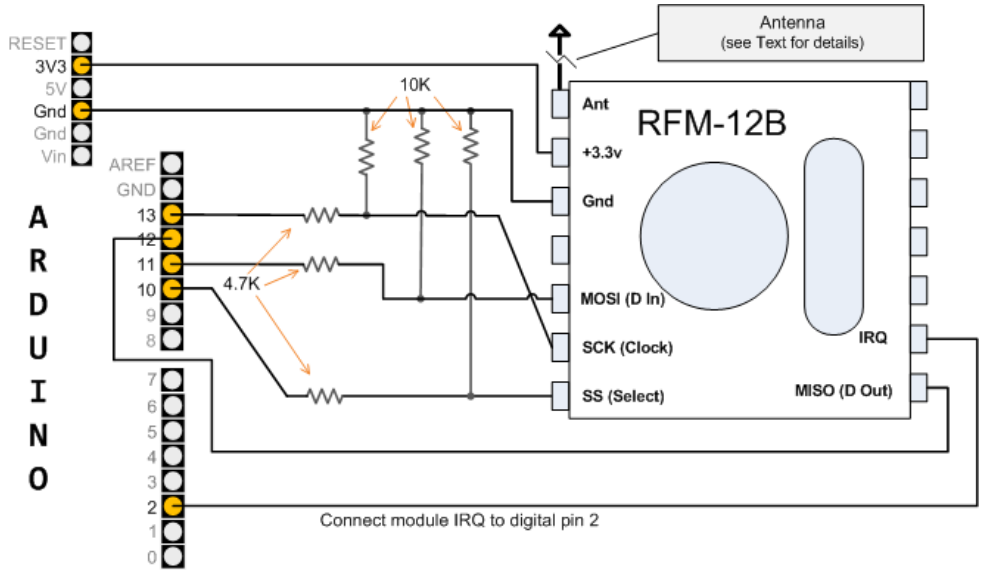

Figure B.1 RFM12B-S2 schematics

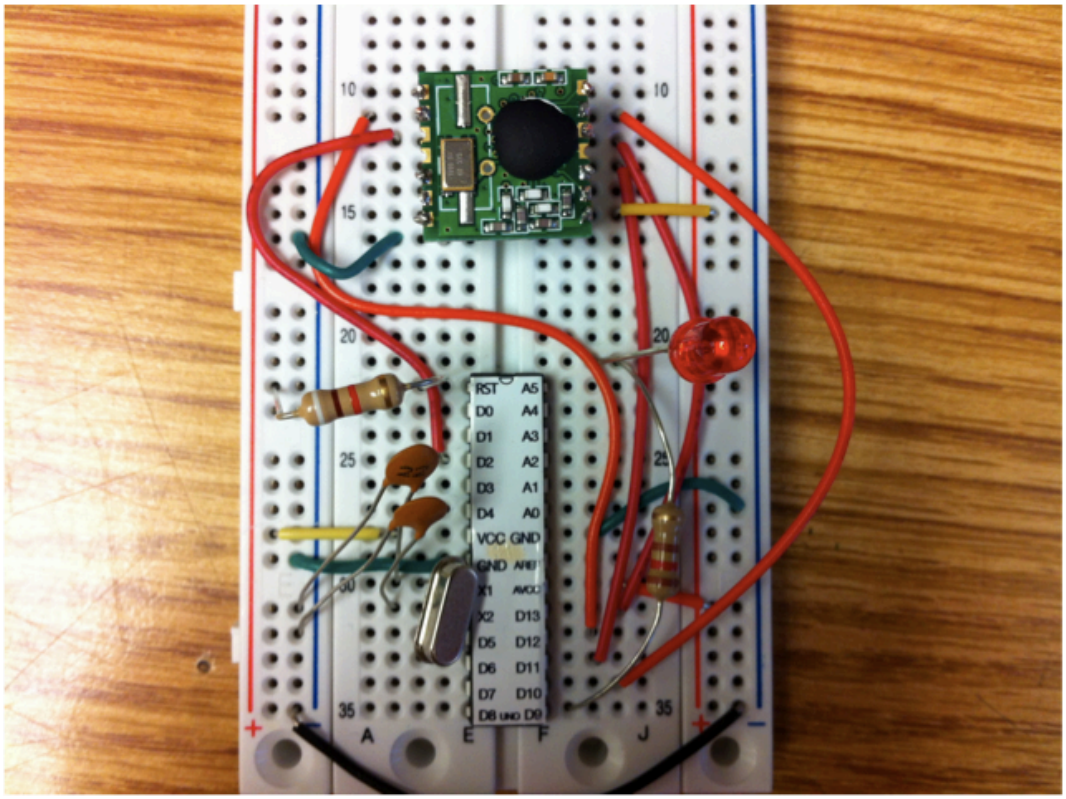

Figure B.2 Wired transceiver with ATmega328

Appendix C Sensor Testing Results

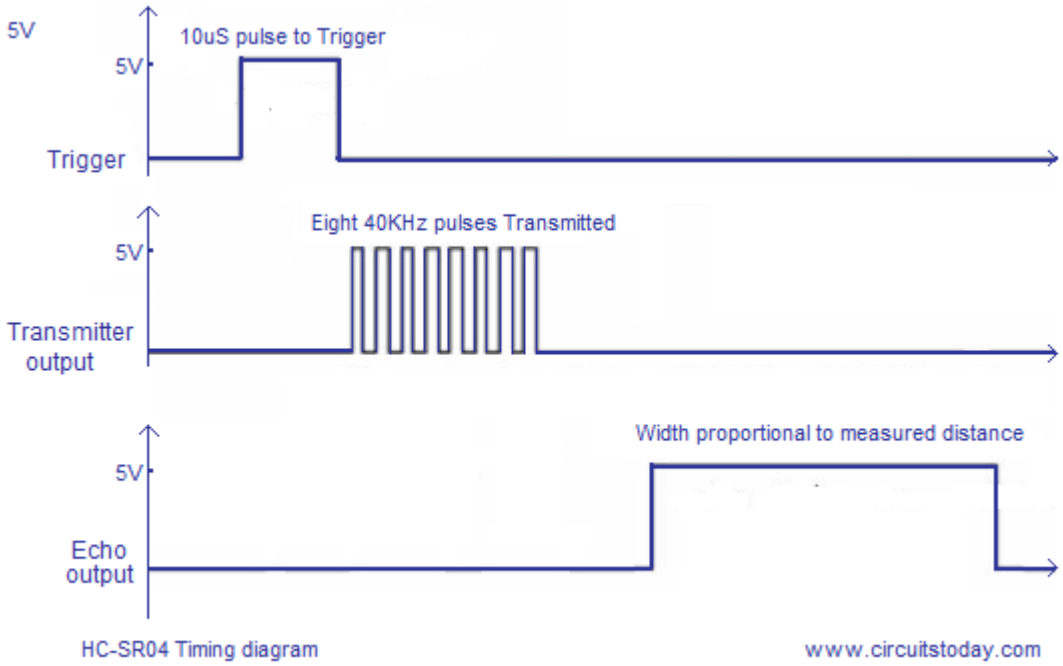

Figure C.1 HC-SR04 timing diagram

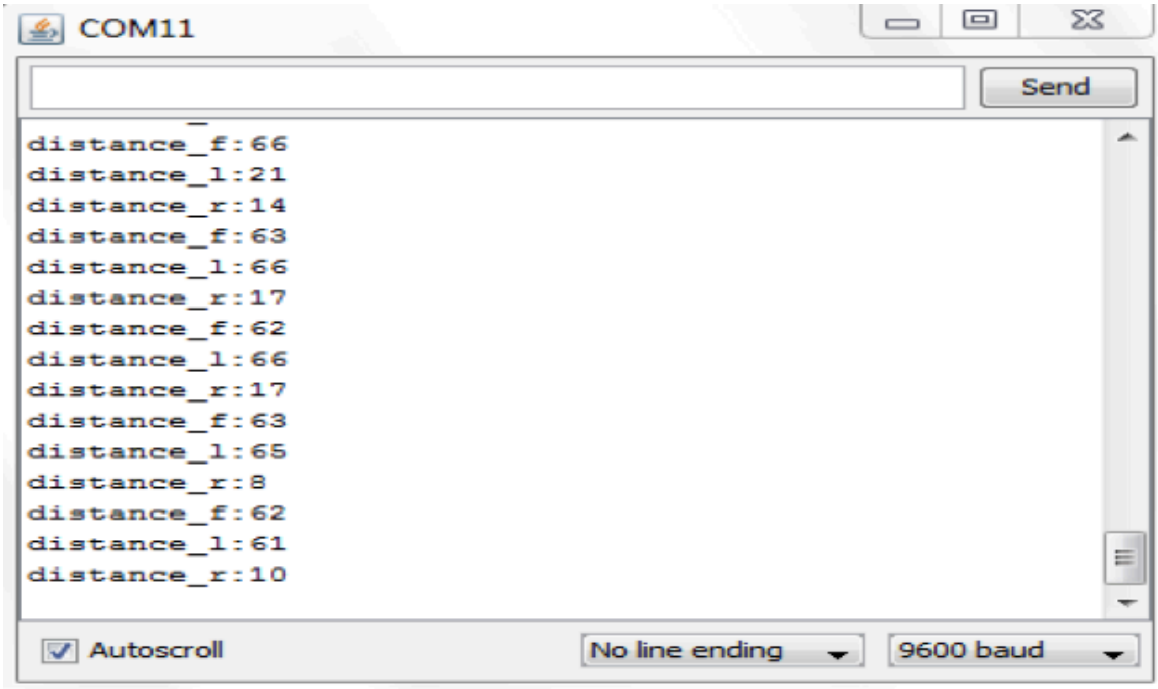

Figure C.2 Sensor Testing results

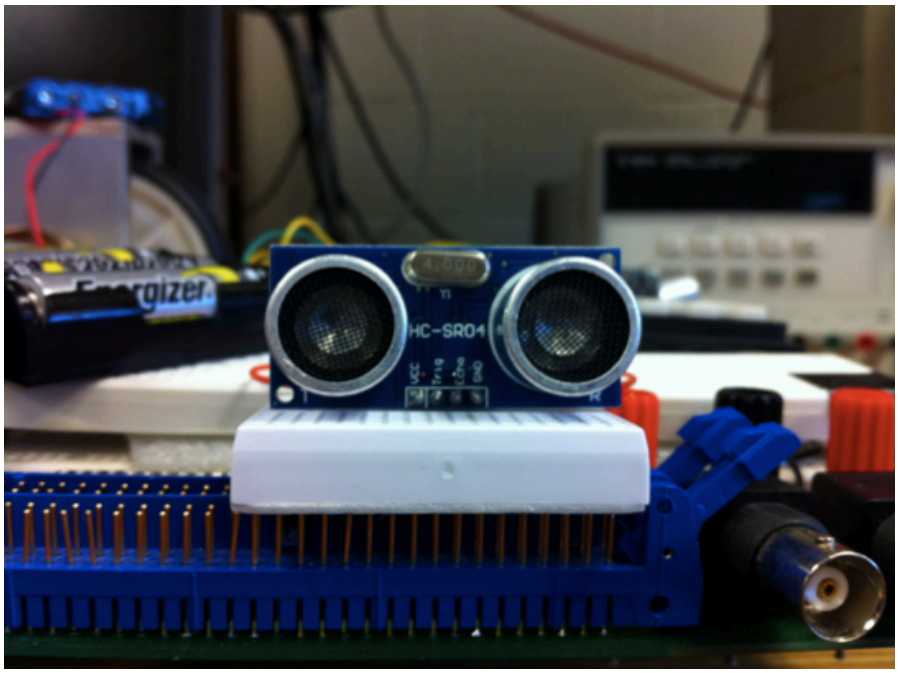

Figure C.3 Sensor Front View

Appendix D Supplemental Materials of User Interface

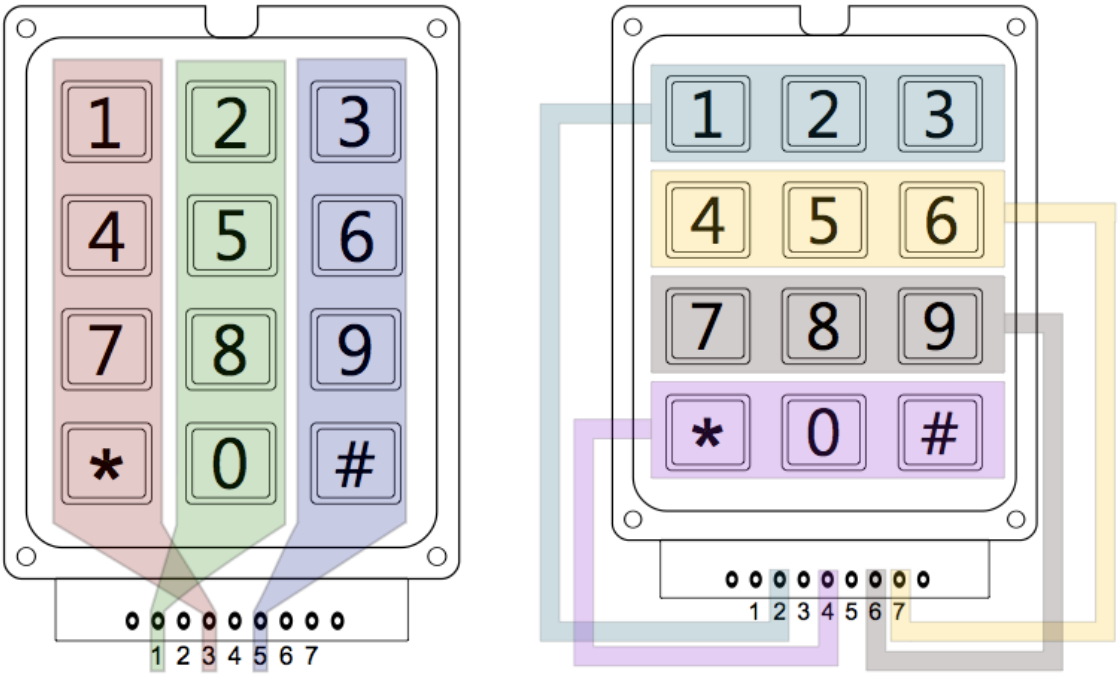

Figure D.1 Keypad pin mapping

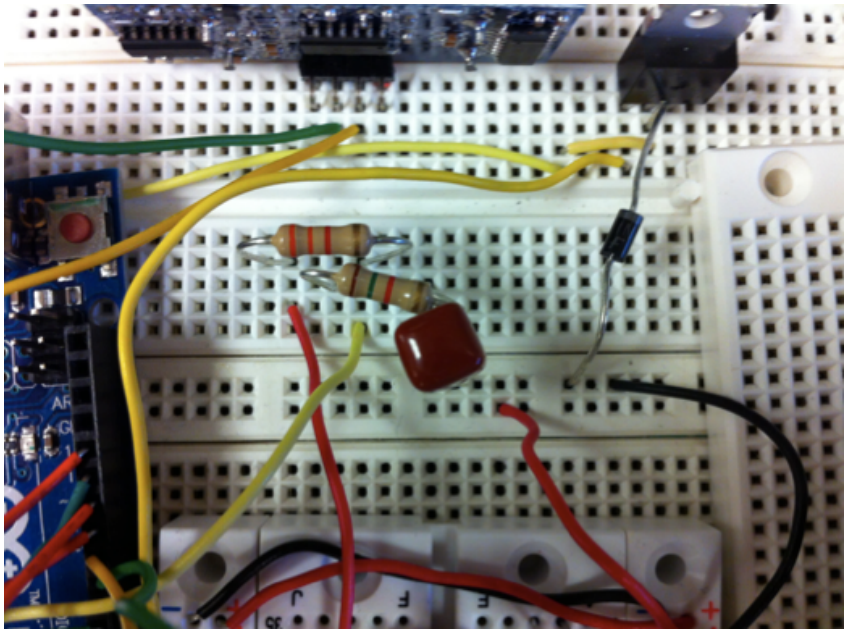

Figure D.2 Low-pass filter

Table D.3 Mapping From ATmega328 to Arduino

Catalogs	Sent From ATmega328 (0 - 255)	Received by Arduino (0 - 1023)
Grocery (0)	23	77
Kids (1)	46	157
Books (2)	69	237
Office (3)	92	317
Health (4)	115	393
Beauty (5)	138	469
Sports (6)	161	543
Electronics (7)	184	615
Apparel (8)	207	683
Home (9)	230	751

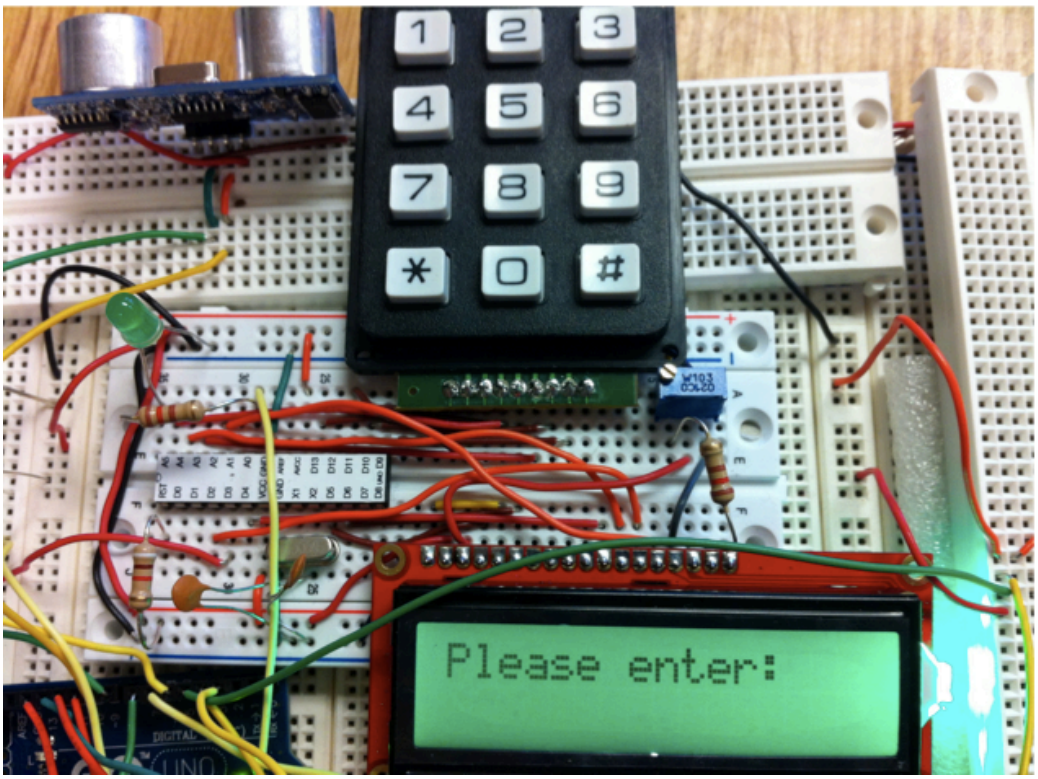

Figure D.4 Top view of user interface

Appendix E Arduino Codes

```
#include <RFM12B.h>
#include <avr/sleep.h>

#define trigPin_l 14
#define trigPin_f 9
#define trigPin_r 4
#define echoPin_l 7 //A0 used as digital input
#define echoPin_f 8
#define echoPin_r 3
#define transistorPin_l 5
#define transistorPin_r 6
#define readPin 3 // read from atmega328 through anaglog pin 3
#define sendreadyPin 15 //A1 used as digital input, connected to atmega

#define NODEID        1 // network ID for customer
#define NETWORKID    99  // the network ID we are on
#define cartID   2  // the node ID we're sending to: human
#define ACK_TIME     50 // wait 50 ms for ACK signal to come back
#define SERIAL_BAUD  9600

int t_07 = 1300; // time needed to cover 0.7 m in x-direction
int t_90deg_left = 1130; //time needed to turn 90 degree when walking
int t_90deg_right = 970; //time needed to turn 90 degree when rest
int t_180 = 2020;
int t_21 = 4050;
int t_14 = 3050;
int t_sensor = 200;
float v_straight = 0.5; //meter per sec
int current_head;

int alarm_f = 0, alarm_l = 0, alarm_r = 0; // 0 if no objects within 35 - 45 cm
float cart_x = 1.3; // store the cart position
float cart_y = 0.5;
int cart_head = 0;
long duration_f, distance_f;
long duration_l, distance_l;
long duration_r, distance_r;
int readytogo = 0; // set to 1 after collecting all store information

// wait for user input when starting up
int sum = 0;
int ave = 0;
int count = 0; // count the numver of items selected
int i = 0;
int j = 0;
```

```
int store[10]; // store the store number
float store_location[10][2]; // store location
//char store_name[10]; // store name

// determine optimal path
int sort = 0; // store list is unsorted, set to 1 if sorted
int goal;
int nextstore = 0;
float next_x;
float next_y;

RFM12B radio;
char alarm[1]; // set to 'Y' if front sensor detects something
bool requestACK = true;

void setup() {
  Serial.begin (9600);
  pinMode(trigPin_f, OUTPUT);
  pinMode(trigPin_l, OUTPUT);
  pinMode(trigPin_r, OUTPUT);
  pinMode(echoPin_f, INPUT);
  pinMode(echoPin_l, INPUT);
  pinMode(echoPin_r, INPUT);
  pinMode(19, OUTPUT); // front sensor LED indicator

  pinMode(sendreadyPin, INPUT);
  pinMode(transistorPin_l, OUTPUT);
  pinMode(transistorPin_r, OUTPUT);

  pinMode(readPin,INPUT);

  radio.Initialize(NODEID, RF12_433MHZ, NETWORKID);
  radio.Sleep(); // sleep right away to save power
  // Serial.println("Transmitting...\n\n");
}

void loop() {
  if(digitalRead(sendreadyPin) == 1){
    delay(500);
    while(i < 10)
    {
      delay(500);
      while(j < 10){
        delay(25);
        int val = analogRead(readPin);
        sum = sum + val;
        j ++;
        //Serial.print("value =");
```

```
  //Serial.println(val);
}
ave = sum / 10; // average out to get user input
//Serial.print("average =");
//Serial.println(ave);
//Serial.print("");


if(ave < 70){
  i --; // no coming signal
}
if(ave < 70 && count != 0){
  break; // no coming signal
}
if(70 < ave && ave < 160){
  store[count] = 0;
  store_location[count][0] = 1.3;
  store_location[count][1] = 0.5;
  //store_name[count] = "Grocery";
  Serial.print("grocery");
  count ++;
}
if(160 < ave && ave < 250){
  store[count] = 1;
  store_location[count][0] = 2.7;
  store_location[count][1] = 0.5;
  //store_name[count] = "Kids";
  Serial.print("kids");
  Serial.print("location:");
  Serial.println(store_location[count][0]);
  Serial.println(store_location[count][1]);
  count ++;
}
if(250 < ave && ave < 340){
  store[count] = 2;
  store_location[count][0] = 2.7;
  store_location[count][1] = 1.9;
  //store_name[count] = "Books";
  Serial.print("books");
  count ++;
}
if(340 < ave && ave < 430){
  store[count] = 3;
  store_location[count][0] = 1.3;
  store_location[count][1] = 1.9;
  //store_name[count] = "Office";
  Serial.print("office");
  count ++;
```

```
}
if(430 < ave && ave < 520){
  store[count] = 4;
  store_location[count][0] = 1.3;
  store_location[count][1] = 3.3;
  // store_name[count] = "Health";
  Serial.print("Health");
  count ++;
}
if(520 < ave && ave < 610){
  store[count] = 5;
  store_location[count][0] = 2.7;
  store_location[count][1] = 3.3;
  //store_name[count] = "Beauty";
  Serial.print("beauty");
  count ++;
}
if(610 < ave && ave < 700){
  store[count] = 6;
  store_location[count][0] = 2.7;
  store_location[count][1] = 4.7;
  //store_name[count] = "Sports";
  Serial.print("sports");
  count ++;
}
if(700 < ave && ave < 790){
  store[count] = 7;
  store_location[count][0] = 1.3;
  store_location[count][1] = 4.7;
  //store_name[count] = "Electronics";
  Serial.print("electronics");
  count ++;
}
if(790 < ave && ave < 880){
  store[count] = 8;
  store_location[count][0] = 2.7;
  store_location[count][1] = 6.1;
  //store_name[count] = "Apparel";
  Serial.print("apparel");
  count ++;
}
if(880 < ave && ave < 970){
  store[count] = 9;
  store_location[count][0] = 1.3;
  store_location[count][1] = 6.1;
  //store_name[count] = "Home";
  Serial.print("home");
  count ++;
```

```
  }

  i ++; // increase the index
  delay(250);
  sum = 0;
  j = 0;
}

delay(1000);
//Serial.print("stores we chose:\n");
//Serial.print("count:");
//Serial.println(count);
// for (int a =0; a < count; a++){
// Serial.println(store[a]);
//Serial.print("");
// }
// determine optimal path
if (sort == 0){
  for(int k = 0; k < count; k ++){
    for(int h = 0; h < count - k -1; h++){
      if(store[h] > store[h+1]){
        int tmp=store[h];
        store[h]=store[h+1];
        store[h+1]=tmp;
        //flip x
        float flip_x = store_location[h][0];
        store_location[h][0] = store_location[h+1][0];
        store_location[h+1][0] = flip_x;
        //flip y
        float flip_y = store_location[h][1];
        store_location[h][1] = store_location[h+1][1];
        store_location[h+1][1] = flip_y;
      }
    }
  }
}


/*Serial.print("sorted stores:");
 for (int b =0; b < count; b++){
 Serial.println(store[b]);
 Serial.print("x_axis:");
 Serial.println(store_location[b][0]);
 Serial.print("y_axis:");
 Serial.println(store_location[b][1]);
 } */
delay(1000);
sort = 1; // finish sorting the list
```

```
  readytogo =1;
 }

 if(readytogo == 1){
  goal = store[nextstore]; // find the next destination

  if (nextstore < count){
   next_x = store_location[nextstore][0]; // get the next store location
   next_y = store_location[nextstore][1];
   nextstore++;
   //test
   Serial.print("next loaction_x:");
   Serial.println(next_x,2);
   Serial.print("next loaction_y:");
   Serial.println(next_y,2);
   delay(1000);
  }
  else {
   next_x = 1.3; // return to starting spot
   next_y = 0.5;
  }

  // set out to the destination!!!
  float delta_x = next_x - cart_x; // calcualate the length needs to go
  float delta_y = next_y - cart_y;

  // U_turn necessary?
  if ((cart_head == 180 && (cart_y == 0.5 || cart_y == 3.3 || cart_y == 6.1)) || (cart_head == 0 &&
(cart_y == 1.9 || cart_y == 4.7))){
    // stop to get ready for U_turn

    // front sensor needed
    digitalWrite(trigPin_f, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin_f, HIGH);
    delayMicroseconds(10); // Added this line
    digitalWrite(trigPin_f, LOW);

    duration_f = pulseIn(echoPin_f, HIGH);
    distance_f = (duration_f/2) / 29.1;
    Serial.print("\ndistance_f: ");
    Serial.println(distance_f);

    while(distance_f < 20 && distance_f > 0){
     digitalWrite(19, HIGH);
     analogWrite(transistorPin_l, 0);
     analogWrite(transistorPin_r, 0);
     delay(150);
```

```
 //talk to cautomer
 radio.Wakeup();

 radio.Send(cartID, alarm, 1, requestACK);
 delay(50); // wait for receive send back ACK

 if (radio.ACKReceived(cartID)){
  Serial.print("successful");
 }

 radio.Sleep();

 delay(500);

 digitalWrite(trigPin_f, LOW);
 delayMicroseconds(2);
 digitalWrite(trigPin_f, HIGH);
 delayMicroseconds(10); // Added this line
 digitalWrite(trigPin_f, LOW);

 duration_f = pulseIn(echoPin_f, HIGH);
 distance_f = (duration_f/2) / 29.1; //update distance front
 Serial.print("new front distance:");
 Serial.println(distance_f);
}
digitalWrite(19, LOW);
analogWrite(transistorPin_r,0);
analogWrite(transistorPin_l,0);
delay(300);

// U_turn
analogWrite(transistorPin_r,0);
analogWrite(transistorPin_l,169);
delay(t_180);
analogWrite(transistorPin_r,0);
analogWrite(transistorPin_l,0);
delay(100);
cart_head = 180 - cart_head; // update the cart head angle

// right sensor needed
digitalWrite(trigPin_r, LOW);
delayMicroseconds(2); // stay low for 2ms
digitalWrite(trigPin_r, HIGH);
delayMicroseconds(10); // stay high for 10 ms
digitalWrite(trigPin_r, LOW);
duration_r = pulseIn(echoPin_r, HIGH);
distance_r = (duration_r/2) / 29.1;
```

```
  if(distance_r < 20 && distance_r > 0){
    analogWrite(transistorPin_l,0);
    analogWrite(transistorPin_r,195);
    delay(t_sensor);
    Serial.print("direction modified!");
  }
}

//return to the origin
if(delta_y < 0){
  if(cart_head == 0){
    analogWrite(transistorPin_r,0);
    analogWrite(transistorPin_l,169);
    delay(t_180); // turn 180 degrees
    analogWrite(transistorPin_r,0);
    analogWrite(transistorPin_l,0);
    delay(300);

    //right sensor needed
    digitalWrite(trigPin_r, LOW);
    delayMicroseconds(2); // stay low for 2ms
    digitalWrite(trigPin_r, HIGH);
    delayMicroseconds(10); // stay high for 10 ms
    digitalWrite(trigPin_r, LOW);
    duration_r = pulseIn(echoPin_r, HIGH);
    distance_r = (duration_r/2) / 29.1;

    if(distance_r < 20 && distance_r > 0){
      analogWrite(transistorPin_l,0);
      analogWrite(transistorPin_r,195);
      delay(t_sensor);
      Serial.print("direction modified!");
    }
    cart_head = 180;
  }
  analogWrite(transistorPin_r,195);
  analogWrite(transistorPin_l,169);
  delay(t_07 * 0.9);
  //turn left 90 degree
  analogWrite(transistorPin_r,195);
  analogWrite(transistorPin_l,0);
  delay(t_90deg_left);
  analogWrite(transistorPin_r,0);
  analogWrite(transistorPin_l,0);
  delay(300);
  analogWrite(transistorPin_r,195);
  analogWrite(transistorPin_l,169);
```

```
 delay((float)(-(delta_y+0.15)/v_straight)*1000);

 if(delta_x ==0){
  //turn left by 90 degree
  analogWrite(transistorPin_r,195);
  analogWrite(transistorPin_l,0);
  delay(t_90deg_left);
  analogWrite(transistorPin_r,195);
  analogWrite(transistorPin_l,169);
  delay(t_07);
  analogWrite(transistorPin_r,0);
  analogWrite(transistorPin_l,0);
  delay(300);
 }
 //turn right by 90 degree
 else if(delta_x < 0){
  analogWrite(transistorPin_r,0);
  analogWrite(transistorPin_l,169);
  delay(t_90deg_right);
  analogWrite(transistorPin_r,195);
  analogWrite(transistorPin_l,169);
  delay(t_07);
  analogWrite(transistorPin_r,0);
  analogWrite(transistorPin_l,0);
  delay(300);
 }
 readytogo = 0;  // stop
}

// go in the x direction for 1.4 m if the destination is in the same lane

else if(delta_y == 0){
 // return from 1 to 0
 if(cart_x == 2.7 && cart_y == 0.5){

  // U_turn
  analogWrite(transistorPin_r,0);
  analogWrite(transistorPin_l,169);
  delay(t_180);
  analogWrite(transistorPin_r,0);
  analogWrite(transistorPin_l,0);
  delay(100);
  cart_head = 180 - cart_head;
  //go for 1.4m
  analogWrite(transistorPin_r,195);
  analogWrite(transistorPin_l,169);
  delay(t_14);
  readytogo = 0;  // stop
```

```
  }
  //go straight for 1.4m
  else{
    //Serial.print("in loop?");
    analogWrite(transistorPin_r,195);
    analogWrite(transistorPin_l,169);
    delay(t_14);
    //Serial.print("done");
  }

  digitalWrite(trigPin_r, LOW);
  delayMicroseconds(2); // stay low for 2ms
  digitalWrite(trigPin_r, HIGH);
  delayMicroseconds(10); // stay high for 10 ms
  digitalWrite(trigPin_r, LOW);
  duration_r = pulseIn(echoPin_r, HIGH);
  distance_r = (duration_r/2) / 29.1;

  if(distance_r < 20 && distance_r > 0){
    analogWrite(transistorPin_l,0);
    analogWrite(transistorPin_r,195);
    delay(t_sensor);
  }
} // delta_y == 0

// the cart needs to move into another lane
else if (delta_y > 0){

  //in x direction 0.7m
  analogWrite(transistorPin_r,195);
  analogWrite(transistorPin_l,169);
  delay(t_07);


  // turn 90 defrees left
  if(cart_y == 0.5 || cart_y == 3.3 || cart_y == 6.1){
    analogWrite(transistorPin_r,195);
    analogWrite(transistorPin_l,0);
    delay(t_90deg_left);
    analogWrite(transistorPin_r,0);
    analogWrite(transistorPin_l,0);
    delay(300);
  }

  // turn 90 degrees right
  else if(cart_y == 1.9 || cart_y == 4.7){
    analogWrite(transistorPin_r,0);
    analogWrite(transistorPin_l,169);
```

```
  delay(t_90deg_right);
  analogWrite(transistorPin_r,0);
  analogWrite(transistorPin_l,0);
  delay(300);

}
// move in y direction

analogWrite(transistorPin_r,195);
analogWrite(transistorPin_l,169);
delay(t_07);

analogWrite(transistorPin_r,195);
analogWrite(transistorPin_l,169);
delay((float)((delta_y) / v_straight) * 1000 - t_07);
Serial.print("time measured:");
Serial.println(delta_y / v_straight * 1000);
Serial.print("delta x:");
Serial.println(delta_x);
Serial.print("cart_head:");
Serial.println(cart_head);

//move in remaining x direction
if(delta_x == 0 && cart_head == 0){
  analogWrite(transistorPin_r,195);
  analogWrite(transistorPin_l,0);
  delay(t_90deg_left);
  analogWrite(transistorPin_r,0);
  analogWrite(transistorPin_l,0);
  delay(300);
  current_head = 180;
}
if(delta_x > 0 && cart_head == 0){
  Serial.print("turn right?");
  analogWrite(transistorPin_r,0);
  analogWrite(transistorPin_l,169);
  delay(t_90deg_right);
  analogWrite(transistorPin_r,0);
  analogWrite(transistorPin_l,0);
  delay(300);
  current_head = 0;
}
if(delta_x == 0 && cart_head == 180){
  analogWrite(transistorPin_r,0);
  analogWrite(transistorPin_l,169);
  delay(t_90deg_right);
  analogWrite(transistorPin_r,0);
  analogWrite(transistorPin_l,0);
```

```
  delay(300);
  current_head = 0;
 }
 if(delta_x > 0 && cart_head == 180){
  analogWrite(transistorPin_r,0);
  analogWrite(transistorPin_l,169);
  delay(t_90deg_right);
  analogWrite(transistorPin_r,0);
  analogWrite(transistorPin_l,0);
  delay(300);
  current_head = 180;
 }//head to the destination
 if(delta_x == 0){

  analogWrite(transistorPin_r,195);
  analogWrite(transistorPin_l,169);
  delay(t_07);

 }
 else if((cart_head == 0 && delta_x > 0) || (cart_head == 180 && delta_x < 0)){

  analogWrite(transistorPin_r,195);
  analogWrite(transistorPin_l,169);
  delay(t_07);

 }
 else if((cart_head == 0 && delta_x < 0) || (cart_head == 180 && delta_x > 0)){

  analogWrite(transistorPin_r,195);
  analogWrite(transistorPin_l,169);
  delay(t_21);
 }

}//delta_y <0
cart_head = current_head;
cart_x = next_x; // update current cart location
cart_y = next_y;

//sensor needed
// front sensor needed
digitalWrite(trigPin_f, LOW);
delayMicroseconds(2);
digitalWrite(trigPin_f, HIGH);
delayMicroseconds(10); // Added this line
digitalWrite(trigPin_f, LOW);

duration_f = pulseIn(echoPin_f, HIGH);
distance_f = (duration_f/2) / 29.1;
```

```
Serial.print("\ndistance_f: ");
Serial.println(distance_f);

while(distance_f < 20 && distance_f > 0){
  digitalWrite(19, HIGH);
  analogWrite(transistorPin_l, 0);
  analogWrite(transistorPin_r, 0);
  delay(150);

  //talk to cautomer
  radio.Wakeup();

  radio.Send(cartID, alarm, 1, requestACK);
  delay(50); // wait for receive send back ACK

  if (radio.ACKReceived(cartID)){
    Serial.print("successful");
  }

  radio.Sleep();

  delay(500);

  digitalWrite(trigPin_f, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin_f, HIGH);
  delayMicroseconds(10); // Added this line
  digitalWrite(trigPin_f, LOW);

  duration_f = pulseIn(echoPin_f, HIGH);
  distance_f = (duration_f/2) / 29.1; //update distance front
}
digitalWrite(19, LOW);
analogWrite(transistorPin_r,0);
analogWrite(transistorPin_l,0);
delay(300);

//right sensor needed
digitalWrite(trigPin_r, LOW);
delayMicroseconds(2); // stay low for 2ms
digitalWrite(trigPin_r, HIGH);
delayMicroseconds(10); // stay high for 10 ms
digitalWrite(trigPin_r, LOW);
duration_r = pulseIn(echoPin_r, HIGH);
distance_r = (duration_r/2) / 29.1;

if(distance_r < 20 && distance_r > 0){
  analogWrite(transistorPin_l,0);
```

```
    analogWrite(transistorPin_r,195);
    delay(t_sensor);
    Serial.print("Right sensor:");
    Serial.println(distance_r);
    Serial.print("direction modified!");
   }

   Serial.print("delay?");
   analogWrite(transistorPin_r, 0);
   analogWrite(transistorPin_l, 0);
   delay(3000);
  }
}
```

Appendix F Supplemental Materials of Ultrasonic Transmitter & Receiver

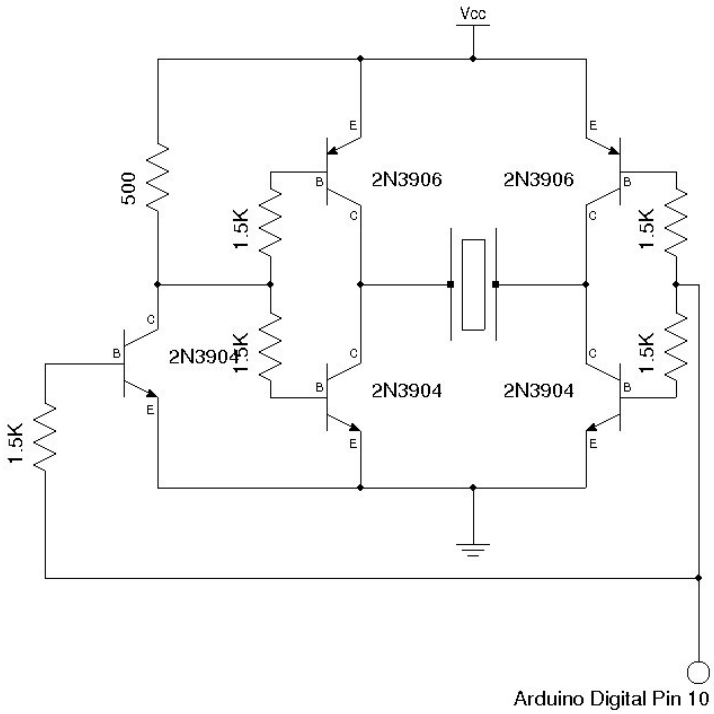

Figure F.1 Circuit diagram of ultrasonic transmitter

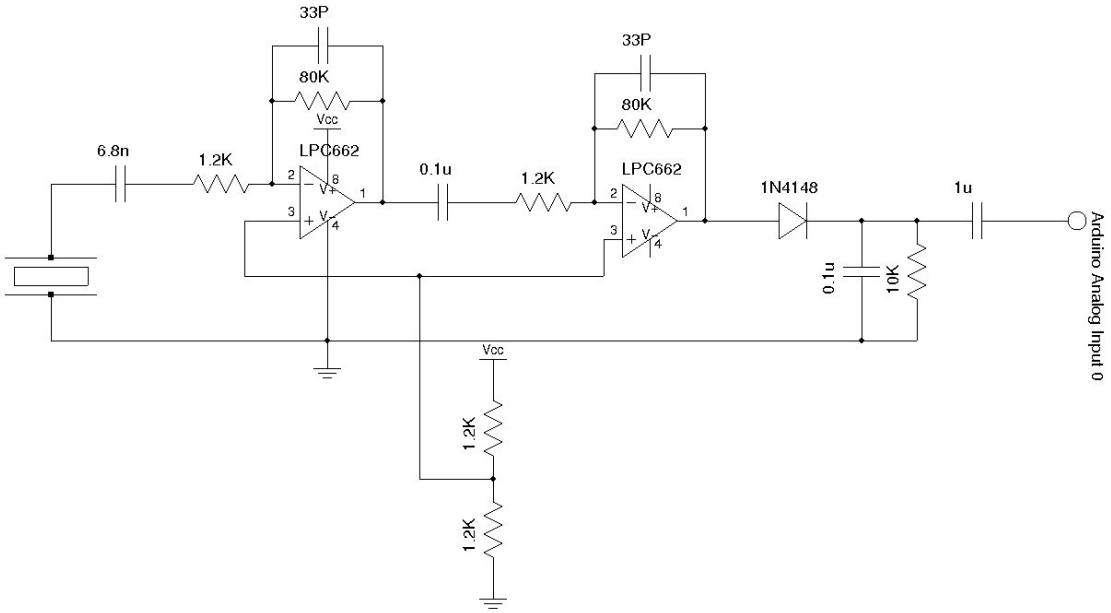

Figure F.2 Circuit diagram of ultrasonic receiver

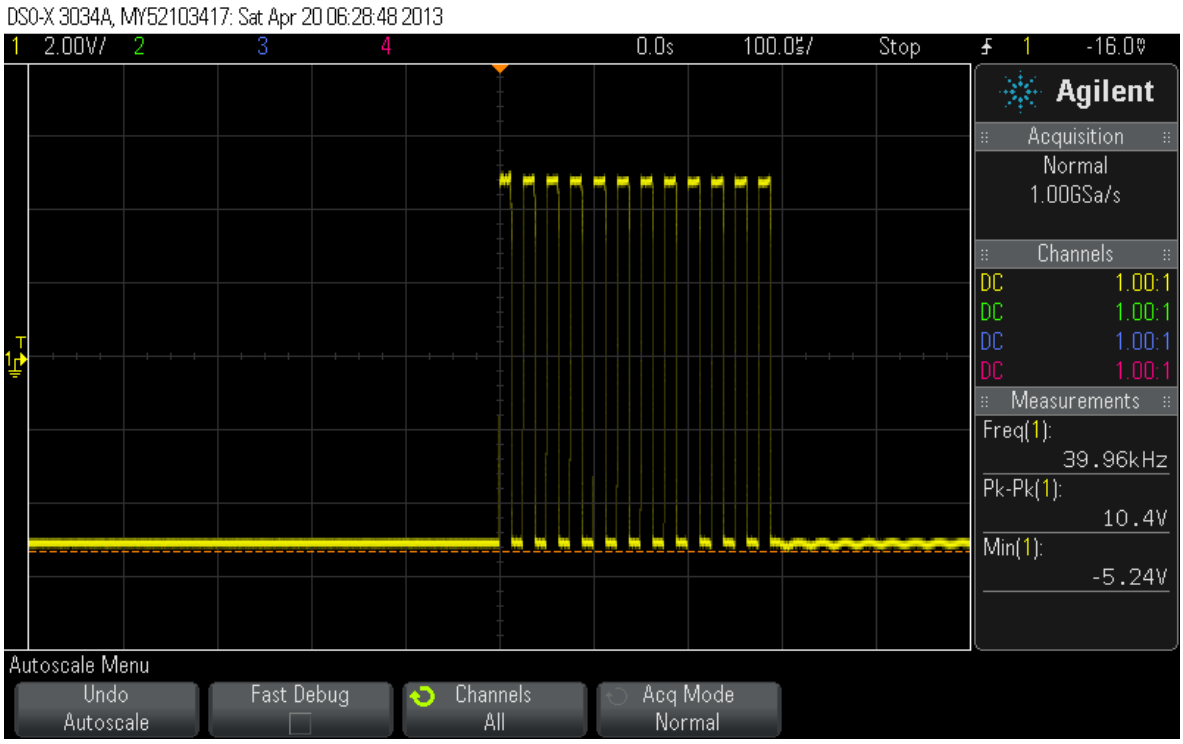

Figure F.3 Signal generated by ultrasonic transmitter

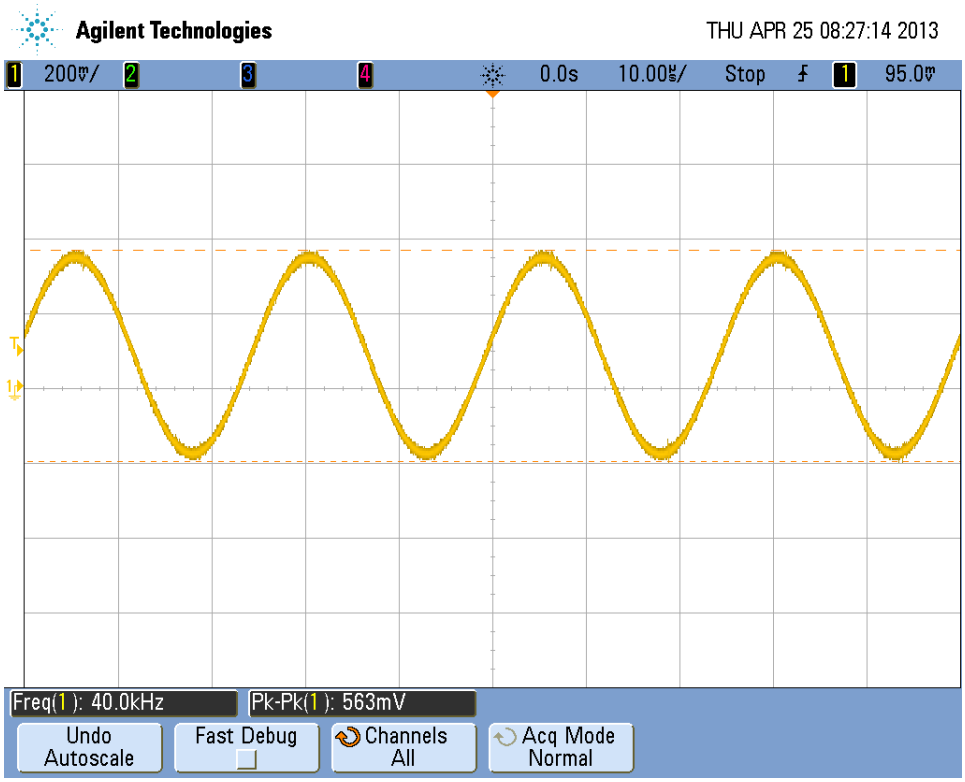

Figure F.4 Signal received by ultrasonic receiver