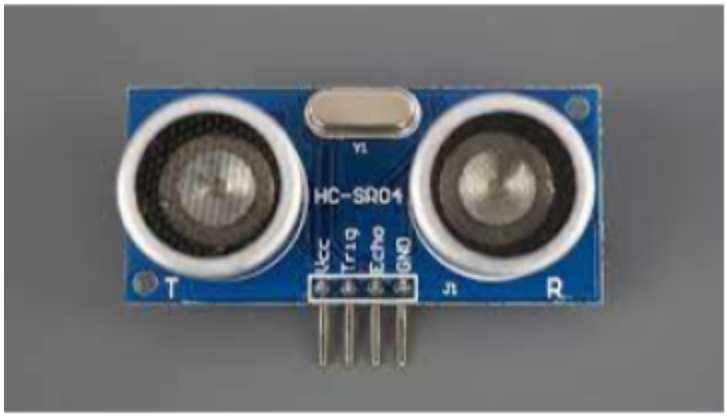
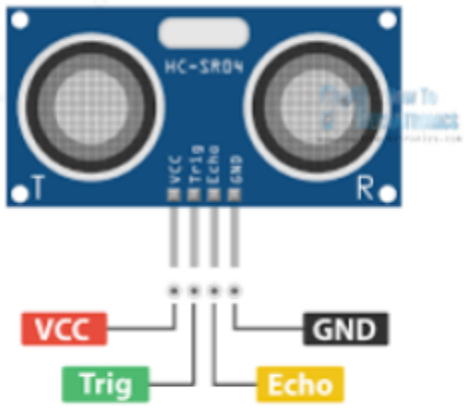
Set up guide for HC-SR04 ultrasonic sensor

Group name: Froshees
Group Members: Sahaj Singh, Sukha Lee, Andrew Speers, Bryce Leung
Course: ENSC 351

This guide will demonstrate how to set up and use the HC-SR04 ultrasonic sensor while connected to the beaglebone green as well as some code to make it work

Table of Contents





Data sheet: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

1.) How the sensor works

The sensor functions by sending a short 10us impulse to the trig pin. This sent impulse tells the sensor to emit an ultrasonic sound that will bounce off the nearest object back to the ultrasonic sensor. We can then measure how long it took for the sound to return and calculate how far away the sensor is using Distance=speed*time. Note that the distance traveled is 2x the distance we want to measure since it travels to the object and back, speed is the speed of sound (343m/s) and time is what we are measuring. The final equation we will use is

$$Distance = (343m/s * time)/2$$

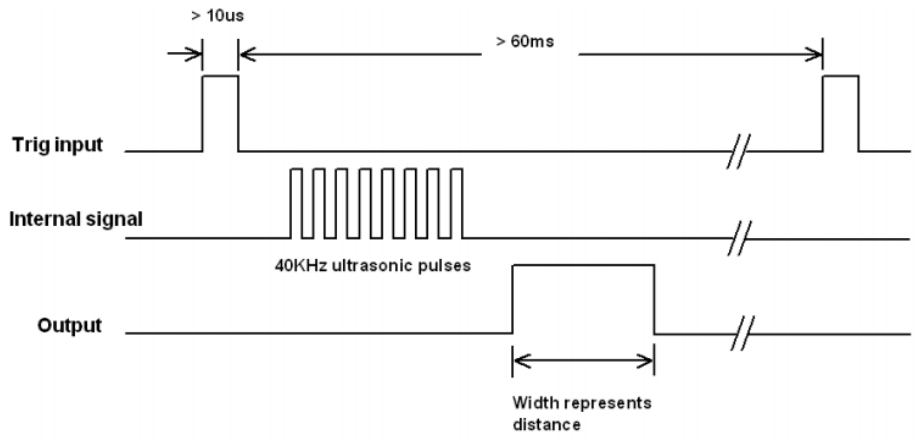


Figure 1.1

2.) Wiring

Required components

- Beaglebone green
- HC SR04 ultrasonic sensor
- Jump wires
- 2k resistor
- 1k resistor

Steps to Prepare the Circuit:

1. Note other resistor values will work just make sure one is twice the value of another
2. Connect the GND of ultrasonic sensor to pin P9 PIN 1 (DGND) directly
3. Connect the VCC of ultrasonic sensor to pin P9 PIN7 (SYS_5V) directly
4. Connect the trig pin of ultrasonic sensor to pin P9 PIN15 (GPIO_48) to directly
5. Connect the echo pin to a 2k resistor that connects to a 1k resistor that connects to pin P9_1 (DGND)
6. Connect pin P9 PIN23 (GPIO 49) in between the 2 resistors

The resistors are needed to reduce the voltage going to the GPIO pin from 5V to under 1.8V that the beaglebone can withstand without damage

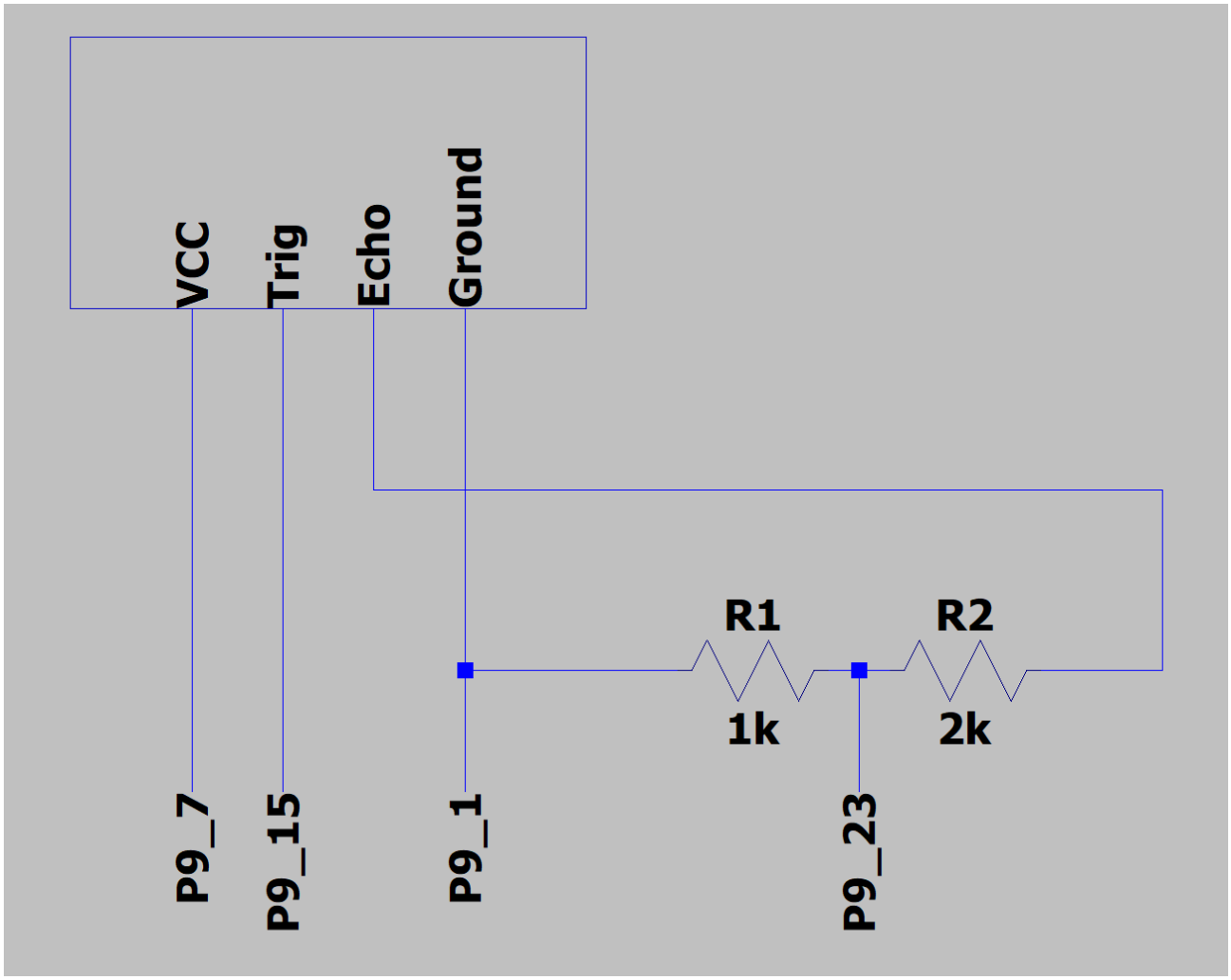


Figure 2.1

3.) Configuring the Beaglebone

SSH into the beaglebone

```
$(bbg) ssh debian@192.168.7.2
```

First we need to make sure that the pins that we are using are set to gpio by running the commands

```
$(bbg) config-pin p9.15 gpio
$(bbg) config-pin p9.23 gpio
```

Check to see if gpio48 and gpio49 are exported

```
$(bbg) ls /sys/class/gpio
```

If gpio48 and gpio49 show up after the ls then the pins are exported
If not go to /sys/class/gpio and export the pins with

```
$(bbg) cd /sys/class/gpio
$(bbg) echo 48 > export
$(bbg) echo 49 > export
```

Now we need to set the pin directions

First the Trig pin

```
$(bbg) cd /sys/class/gpio/gpio48
$(bbg) echo out > direction
$(bbg) cat direction
```

Second the Echo pin

```
$(bbg) cd /sys/class/gpio/gpio49
$(bbg) echo in > direction
$(bbg) cat direction
$(bbg) echo 0 > active_low
$(bbg) cat active_low
```

The beaglebone pins should now be set up

Troubleshooting

1) If you get the error -*bash: echo: write error: Operation not permitted* when exporting then the pin was most likely already exported and you may have unexported it check to see if gpio48/gpio49 are under /sys/class/gpio and if they aren't run the commands

```
$(bbg) echo 48 > export
$(bbg) echo 49 > export
```

   For each that aren't there

4.) Coding the Ultrasonic Sensor in C

The following is a preview of code that can control the ultrasonic sensor using C
See supporting files for full code

pulse_loop function

This function is used to create a thread that sends a 10 microsecond burst to the ultrasonic sensor 5 times a second to collect data that is used by get_data _n_cm

Get_distance_cm

This function calculates how far away the nearest object is. Reads the value of the gpio pins and records when the values are changed and how long it took to obtain the time interval. It then multiplies that time interval by 0.000017150 which is half the speed of sound scaled from ns to seconds. This ends up giving us the distance we need in cm.

```
//Include Headers
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <time.h>
#include <stdbool.h>
#include <pthread.h>

//Definitions
#define Trig_value "/sys/class/gpio/gpio48/value"
#define Echo_value "/sys/class/gpio/gpio49/value"

//used to create a thread that sends an impulse every 0.2s
void* pulse_loop(void* input)
{
   while (true)
   {
      sleepForMs(200);
      writeIntToFile(Trig_value,1);
      sleepForMs(0.01);
      writeIntToFile(Trig_value,0);
   }
   return NULL;
}

// returns how far away the target is from the sensor in cm
//assumed using pin 9_ 23 for Echo and pin 9_15 for Trig
long double get_distance_cm()
{
   long double start_time=0;
   long double end_time=0;
   long double length_of_time=0;
   long double distance_in_cm=0;
```

```
    while(getDataFromFile(Echo_value)==48)//ascii 0
        start_time = getTimeInNs();

    while(getDataFromFile(Echo_value)==49)//ascii 1
        end_time = getTimeInNs();

    length_of_time = end_time-start_time;
    distance_in_cm = length_of_time*0.000017150;//convert to cm
    if (distance_in_cm<=0)
        distance_in_cm=0;
    printf("distance: %.1Lf cm \n",distance_in_cm);
    return distance_in_cm;
}
```

Troubleshooting

1) Make sure you cross compile the code to your beaglebone if needed use the following command to compile it:

```
all:
arm-linux-gnueabihf-gcc -Wall -g -std=c99 -pthread -D _POSIX_C_SOURCE=200809L sensor_test.c
-o sensor_test

cp sensor_test $(HOME)/cmpt433/public/myApps/
```

2) If when running the code the screen doesn't display distances then make sure all pins are exported and the directions are set from part 3