This write up is meant to describe how you might use an acoustic ranging module to measure distance to an object. First the module, shown in Figure 1., is available at

http://www.mpja.com/Ultrasonic-Ranging-Raspberry-Pi-Arduino-Compatible-Module/productinfo/19605+UT/

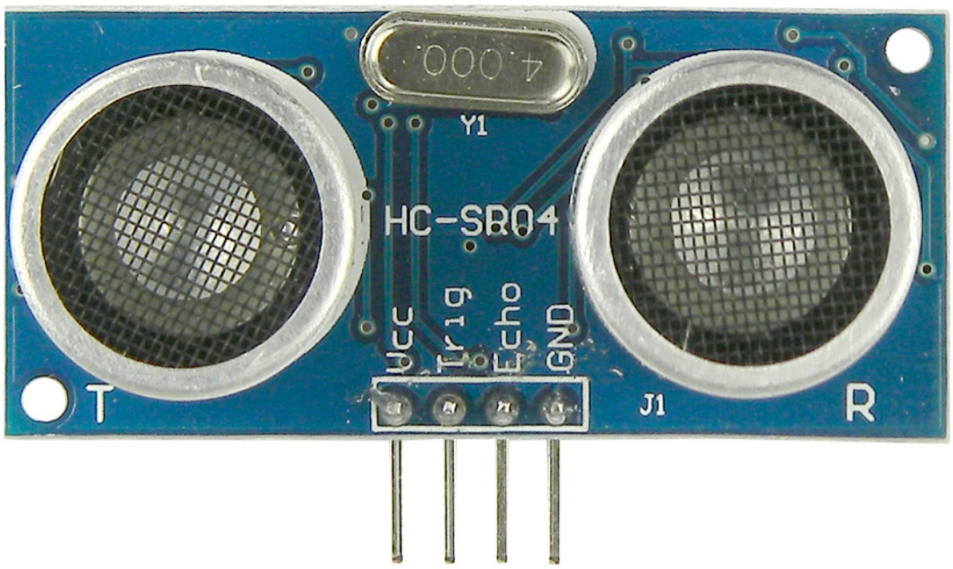

Figure 1. Acoustic Ranging Module

The pin out can be seen in Figure 1., but what does each mean? The pin marked Vcc is the power or 5 volts, and GND is the ground. In order to figure out the use of Trig (short for trigger) and Echo, we refer to the timing diagram in Figure 2. In this Figure we will be sending Trigger (Trig) to the module to start the ranger sampling the distance. Then the time between the end of the burst, sent to the transmitter T, and the echo, received at R, is the travel time of the sound. A basic equation for computing the distance is given in Figure 2.

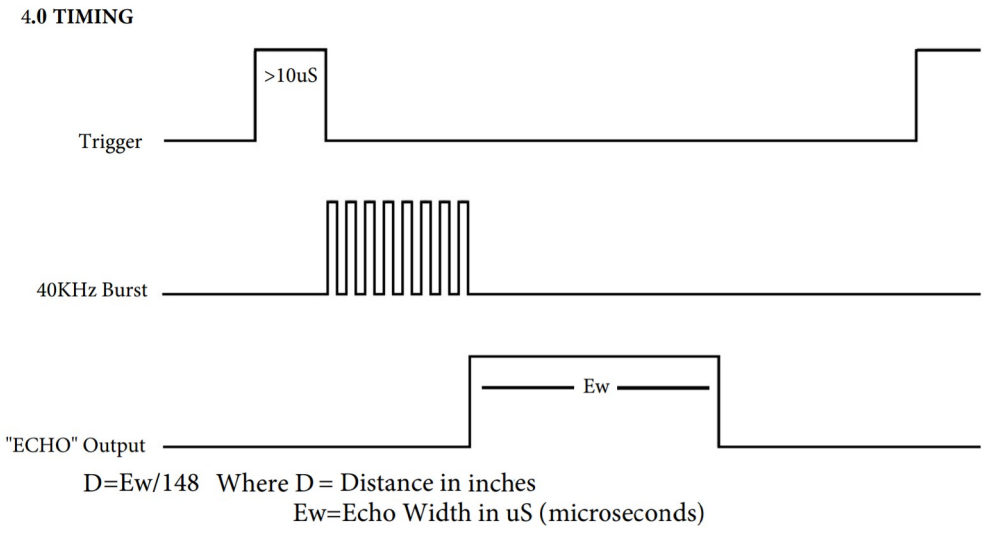

Figure 2. Timing of Signals Trig and Echo

In order to measure the time somewhat close to the microsecond level, we will need to break the rule of not stalling the system waiting for a signal. But remember the grammar rule, you can break any rule as long as you know the rule, and you know why you are breaking it. In this case we are stalling the system (waiting for the Echo signal to go high, and then waiting for it to go low.

```
// Perform Measurement
digitalWrite(TriggerPin, HIGH);
MeasurementTimer = micros();
// Stall for 10 micro
while (micros() - MeasurementTimer < 10);
// Set Trigger low
digitalWrite(TriggerPin, LOW);
// wait for Echo high
while (digitalRead(EchoPin) == LOW);
// Time when Echo went high.
MeasurementTimer = micros();
// Wait for Echo to go low. (Stalls as long as it is high ).
while (digitalRead(EchoPin) == HIGH);
// Compute travel time
MeasurementTimer = micros() - MeasurementTimer;
// Compute distance in inches.
return (float)MeasurementTimer / 148.0;
```

A text box was added to the RoboRemo interface, given an id of D (for distance) and is shown in Figure 3.

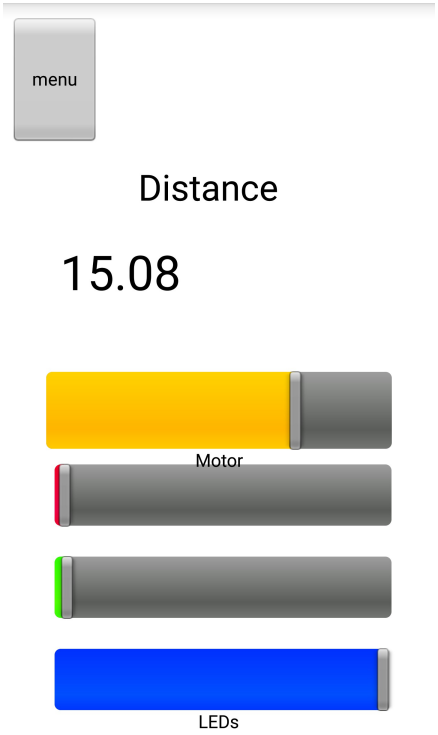

Figure 3. RoboRemo Interface.

Appendix A: Code for Controlling Motor and Colored LED

```
#include <Stepper.h>

// Setup stepper motor controller.
Stepper StepperOne(100, A0, A1, A2, A3);

// Stepper Motor motion parameters
int StepperLocation = 0;     // Current location Arbitrary
int StepperDestination = 0; // Desired Destination
int StepInput, StepperMoving = 0; // Variables for input and feedback.
int Reset = 0;

// function to handle moving stepper forward.
void StepperMove()
{
       // if destination is lower than location
       if (StepperLocation > StepperDestination)
       {
              if (digitalRead(12) == 0 // if limit indicator is active
                     && !Reset)           // but not trying to correct.
              {
                     // Set Destination as two steps back.
                     StepperDestination = StepperLocation + 3;
                     Reset = 1;
              }
              else // Take a negative step.
              {
                     StepperOne.step(-1);
                     StepperLocation--;
              }
       }
       else if (StepperLocation < StepperDestination) // if opposite
       {
              if (digitalRead(12) == 0 // if limit indicator is active
                     && !Reset)            // but not trying to correct.
              {
                     // Set Destination as two steps back.
                     StepperDestination = StepperLocation - 3;
                     Reset = 1;
                     //StepperOne.step(-2);
              }
              else // Take a positive step.
              {
                     StepperOne.step(1);
                     StepperLocation++;
              }
       }
       else // if we are at our destination
       {
              if (StepperMoving)
              {
                     Serial.println("At Destination");
                     StepperMoving = 0;
                     Reset = 0;
              } // End of feedback if

       } // End of Pos. Neg. at destination if
} // End of StepperMove
```

```
#define STEP_INTERVAL 20
unsigned long StepTimer;
unsigned long Timer;
unsigned long CurrentStepInterval = 200; // Initial step time.

// Simple timer to flash LED.
#define LED_INTERVAL 500
unsigned long LedTimer;

#define RedLed   11
#define GreenLed 10
#define BlueLed   9

// Uses Acoustic sensor to measure distance.
#define EchoPin A5
#define TriggerPin A4
unsigned long MeasurementTimer;
float MeasureDistance()
{
       // Perform Measurement
       digitalWrite(TriggerPin, HIGH);
       MeasurementTimer = micros();
       // Stall for 10 micro
       while (micros() - MeasurementTimer < 10);
       // Set Trigger low
       digitalWrite(TriggerPin, LOW);
       // wait for Echo high
       while (digitalRead(EchoPin) == LOW);
       // Time when Echo went high.
       MeasurementTimer = micros();
       // Wait for Echo ot go low. (Stalls as long as it is high ).
       while (digitalRead(EchoPin) == HIGH);
       // Compute travel time
       MeasurementTimer = micros() - MeasurementTimer;
       // Compute distance in inches.
       return (float)MeasurementTimer / 148.0;
} // End of MeasureDistance

// Character string and pointer for incoming characters.
char IncomingChar[128];
int  IncomingPointer = 0;

// Run once, To set up system
void setup()
{
       // Sets time of individual steps.
       StepperOne.setSpeed(30);

       // Initialize LED timer and pin
       LedTimer = millis();
       pinMode(13, OUTPUT);

       // Set up Acoustic distance measurement pins
       pinMode(TriggerPin, OUTPUT);
       pinMode(EchoPin, INPUT_PULLUP);

       // Set up Tricolor LED.
       analogWrite(RedLed, 100);
       analogWrite(GreenLed, 0);
       analogWrite(BlueLed, 0);

       // Serial Com.
       Serial.begin(9600);
```

```
	// Initialize motor to close to interrupter.
	// Move stepper until sensor low
	while (digitalRead(12))
	{
		StepperOne.step(-1);
	} // Move to sensor low position

	// Move back until sensor high
	while (!digitalRead(12))
	{
		StepperOne.step(1);
	} // Move to zero position

} // End of setup

// Code that is run continuously.
void loop()
{
	char Incoming;
	int IncomingValue;

	// LED Flashing Timer (~1 second).
	if (millis() - LedTimer >= LED_INTERVAL)
	{
		// Toggle LED.
		if (digitalRead(13))
		{
			// Send to RoboRemo.
			Serial.print("D ");
			Serial.print(MeasureDistance());
			Serial.print("\n");
			digitalWrite(13, LOW);
		}
		else
		{
			digitalWrite(13, HIGH);
		}
		LedTimer += LED_INTERVAL; // Update Timer.

	} // End of Led Timer.

	// Stepper motor Timer.
	if (millis() - StepTimer >= CurrentStepInterval)
	{
		StepTimer = millis(); // Update timer.
		// Yes it is different than we usually do.
		// But due to the varying time interval,
		// this works more consistently.

		StepperMove(); // Update stepper.
		// Adjust Stepper time interval, ramping up ,then down
		if (abs(StepperLocation - StepperDestination) > 10)
		{
			if (CurrentStepInterval > 20) // Provided we are not moving to fast.
			{
				// Speed up as we get to moving.
				CurrentStepInterval -= 20;
			}
		}
		else // We are getting close to our destination
		{
			// Make interval larger until it is 200 milliseconds.
			if (CurrentStepInterval < 200)
				CurrentStepInterval += 20;
		}
	} // End of Stepper Timer if.
```

```
	// Check for incoming serial data.
	if (Serial.available())
	{
		Incoming = Serial.read();

		// Decode incoming serial data.
		if (Incoming != '\n') // If not at terminator
		{
			// place character in buffer.
			IncomingChar[IncomingPointer++] = Incoming;
			// Move pointer ahead one.
		}
		else // With terminator in.
		{
			IncomingChar[IncomingPointer] = 0; // add terminator to string.
			// Decode Results, starting with the first character.
			switch (IncomingChar[0])
			{
			case 'M': // Set motor position
				StepperDestination = atoi(&IncomingChar[1]);
				break;
			case 'R': // Set Red Led
				IncomingValue = atoi(&IncomingChar[1]);
				analogWrite(RedLed, IncomingValue);
				break;
			case 'G': // Set Green Led
				IncomingValue = atoi(&IncomingChar[1]);
				analogWrite(GreenLed, IncomingValue);
				break;
			case 'B': // Set Blue Led
				IncomingValue = atoi(&IncomingChar[1]);
				analogWrite(BlueLed, IncomingValue);
			} // End of decode switch

			IncomingPointer = 0; // Reset pointer

		} // End of Terminator check

	} // End of Serial available if

} // End of loop
```