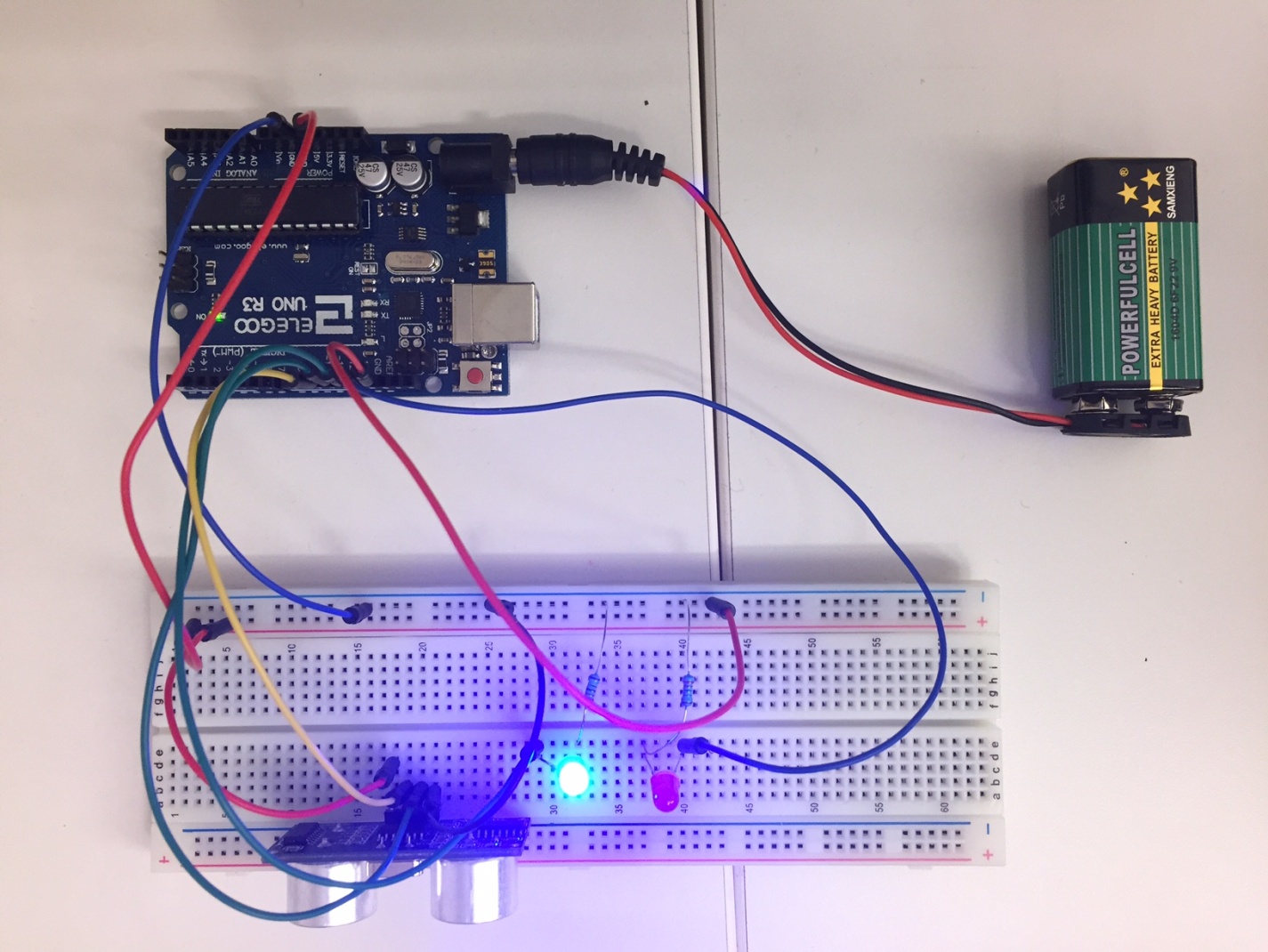
**Ultrasonic Distance Detector for Arduino**

By **Patrick Cook (C/GCC/ASM/Testing)**,

**Lucas Anesti (Project Lead/C/GCC/Testing)**

**Nicholas Hersey (Report/Research)**

****

*This is a snapshot of the device that is able to detect distances and flash results.*

**Table of Contents**

**Table of Contents** – Page 2

**Introduction** – Page 3

**The Microcontroller Platform** – Page 4

**The Test Device** – Page 5

**Development Tools** – Page 6

**Experiment** – Page 7 - 8

**Conclusion** – Page 9

**Contributions** – Page 10

**Project Code** – Page 11

**Introduction**

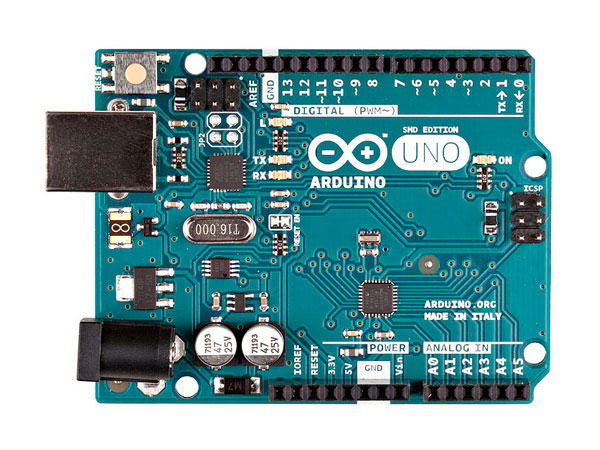
For this project, an Arduino Uno will be used to process information from an ultrasonic sensor that measures distance. Two Led Lights will be used for flashing the output, distance data that the Arduino processed. A 5V battery was used to power this device along with various wires to connect all the parts of the project.

**The Microcontroller Platform**

For this project, the Arduino Uno will be used to process information from the Ultra Sonic Sensor and to the Led Lights.

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

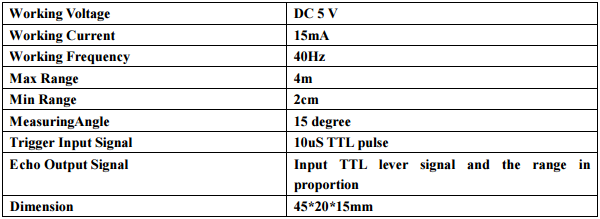


Source: <http://www.arduino.org/products/boards/arduino-uno>

**The Test Device**

For this project, the HC-SR04 Ultrasonic Sensor will be used to send and receive ultra sonic signals that will be timed to judge distance. The HC-SR04 Ultrasonic Sensor has problems with accurately reading non-flat sided objects or scenes, so the device is best at reading the distance of flat sided objects or detecting differences in the scene at least. This sensor can be used to automatically turn on lights for a user to potentially save power.

**Specifications**



Source: <http://www.micropik.com/PDF/HCSR04.pdf>

**Development Tools**

For this project, C/GCC code was used with the Arduino IDE to make the test the first version of the project then the .elf file was used to debug timing issues and then an assembly file using AVR-GCC was created. All coding was done in the Linux environment.

**Experiment**

This device detects distances using the ultrasonic sensor and flashes LED lights along with a duo-tone to output distance data. For this device to carry out its mission it will send an echo and receive an echo back. Since sound travels at a certain predictable speed, half of the distance the echo traveled will be the distance of the object. This distance of the echo can be found by multiplying speed by time.

To read the device, wait for both lights to blink at the same time after pointing the device at an object one would like to read the distance of. After both lights blink, then each blue light means an additional foot, each red light means an additional inch, when both lights blink again then the result is done being expressed.

The device has certain ports it connects to. Below is the pin out for the project, parenthesis is the Arduino pin number:

VCCPin -----> 5V

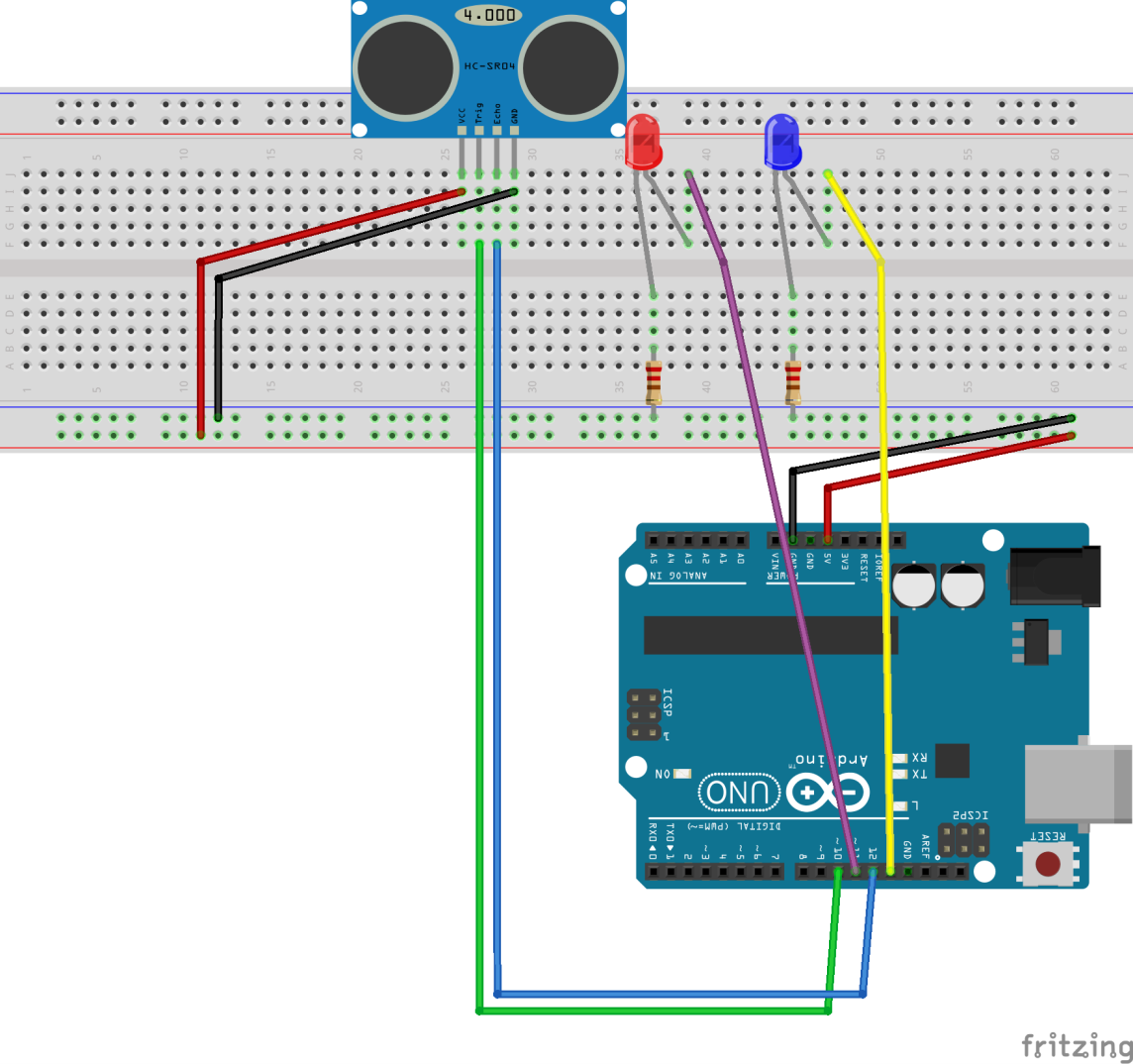
Trigger Pin -----> PB2 (10)

Echo Pin -----> PB4 (12)

GND Pin -----> GND

Red LED (LED\_PIN) -----> PB5 (13)

Blue LED (LED\_PIN2) -----> PB3 (11)



**Conclusions**

This experiment clarified the nature of how these types of devices work. The signals for this device are paired for each pin, each signal per pin usually means off or on, or some sort of status or not-of-status. It also shows that programming isn't just programming, but also a bit of science as well, science that isn't very hard. It just shows that whatever seems difficult to make isn't, it is just complex to some degree.

**Contributions**

Patrick Cook – C/AVR Assembly Coder

Lucas Anesti – Project Lead/Testing/C

Nicholas Hersey – Report Creator/Research

**Project Code**

All project code can be found in the Team Project folder and is separated into an assembly folder (ASM\_RangeFinder) and a C folder (C\_RangeFinder). The standard Makefiles are included in the includes folder.