ROB - Sensors lab

Adam Fabo

Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2. 612 66 Brno - Královo Pole

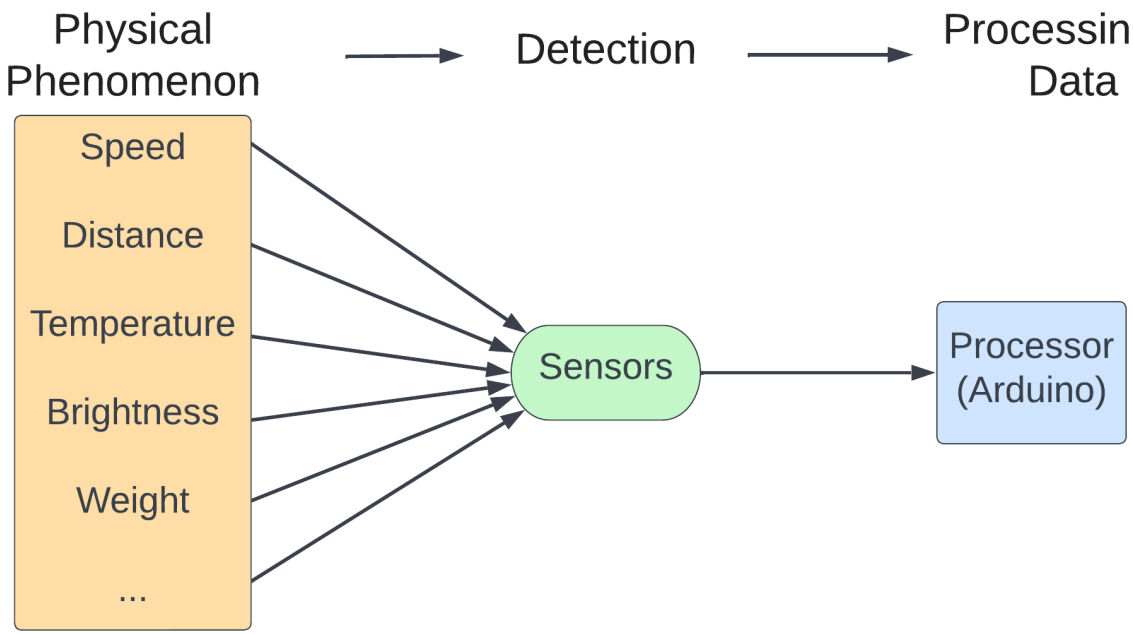
xfaboa00@fit.vutbr.cz

February 6, 2022

Today's laboratory will be focused on sensors.

What are sensors used for

Sensors are used for observing real life physical phenomenon and transforming it into electrical - processor readable way.

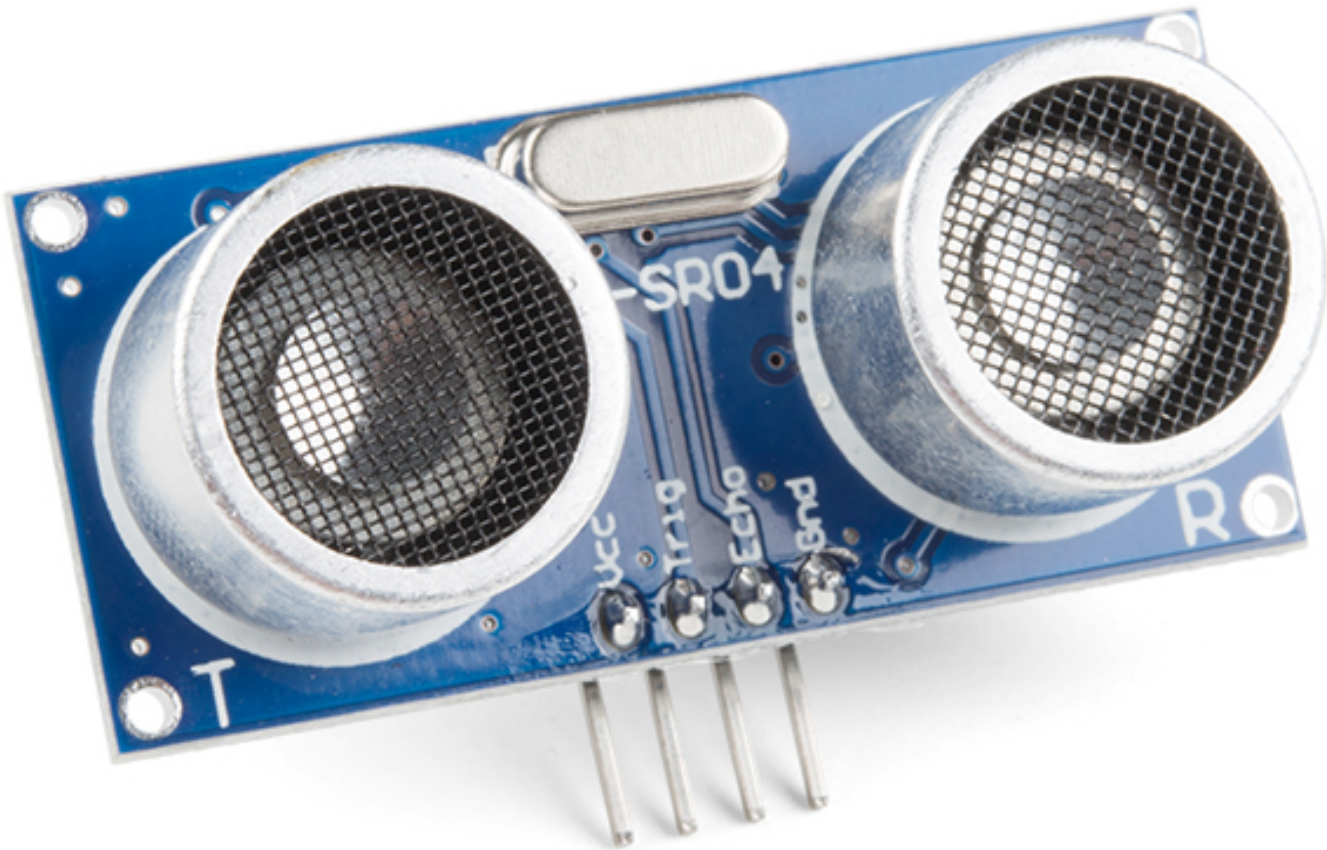


Sensors used in this lab

Sensor that you will use in this lab are the following:

HC-SR04	Ultrasonic distance sensor
SRF08	Ultrasonic distance sensor that uses I2C
minIMU-v5	Inertial Measurement Unit that uses I2C

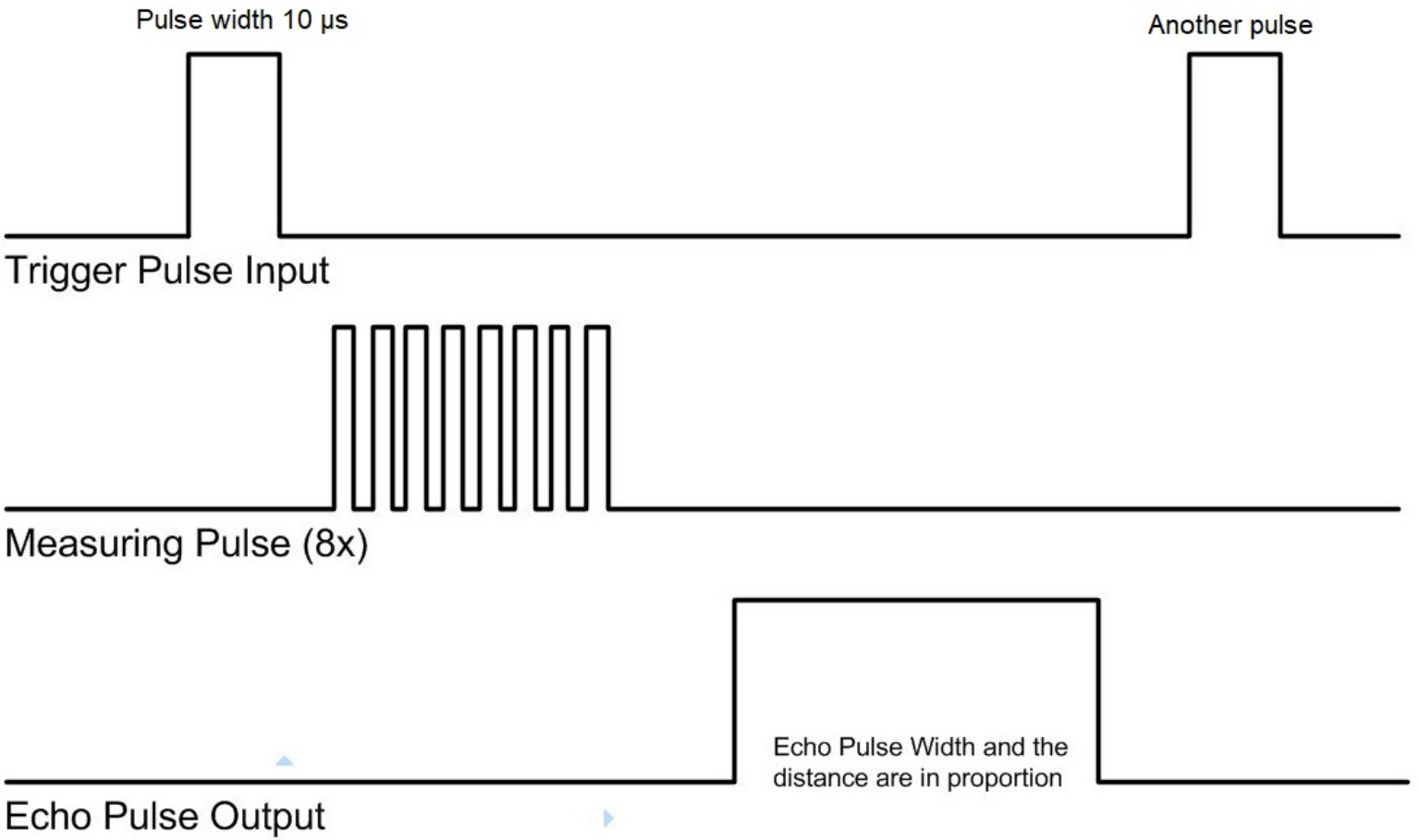
HC-SR04 - Ultrasonic Distance Sensor



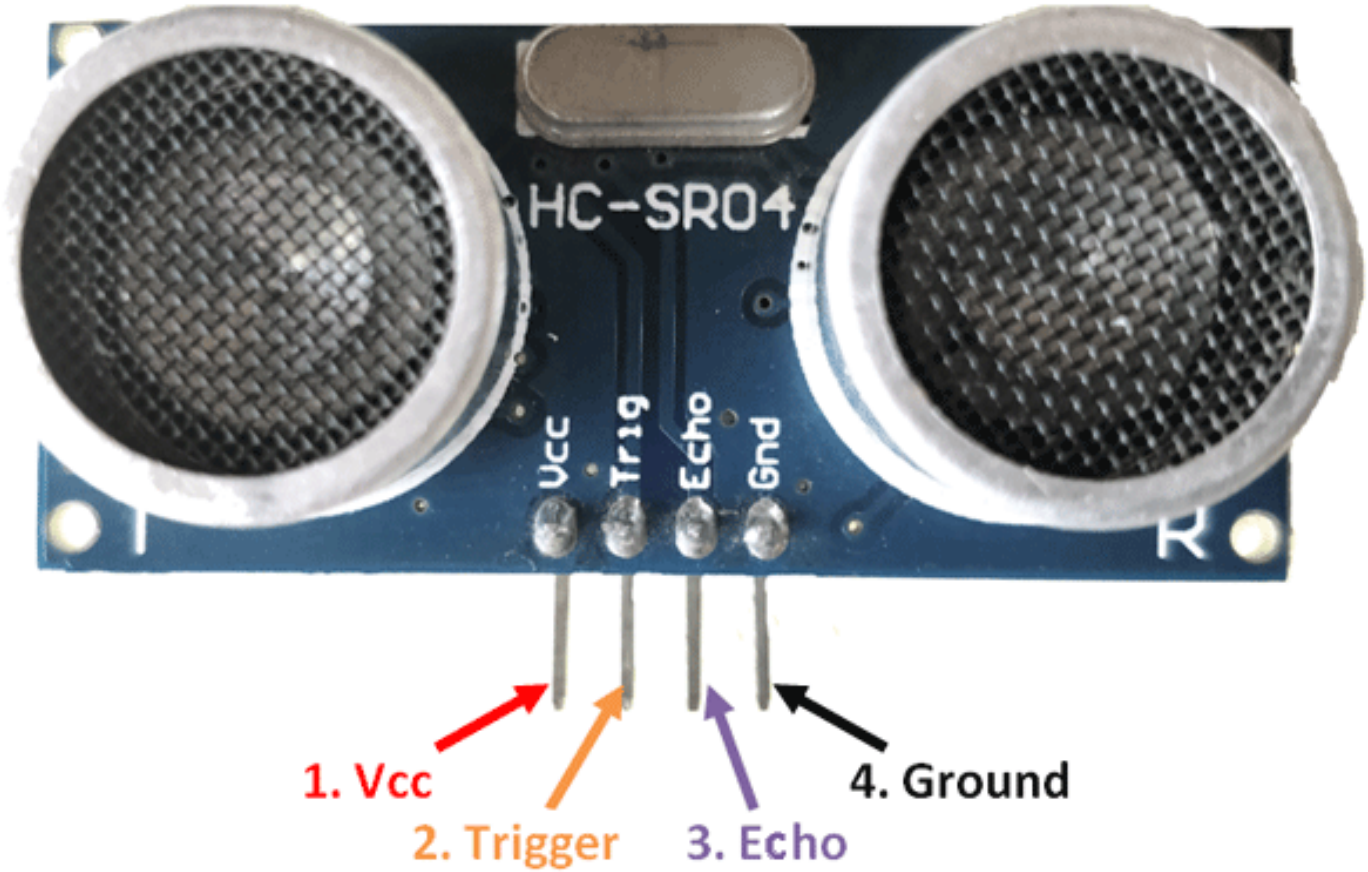
HC-SR04 - How it works?

- Sound waves are fired from the sensor.
- Waves get reflected from objects.
- Sensor can detect those reflections and time it took.
- From the time, using speed of sound distance can be calculated.

HC-SR04 timing diagram:



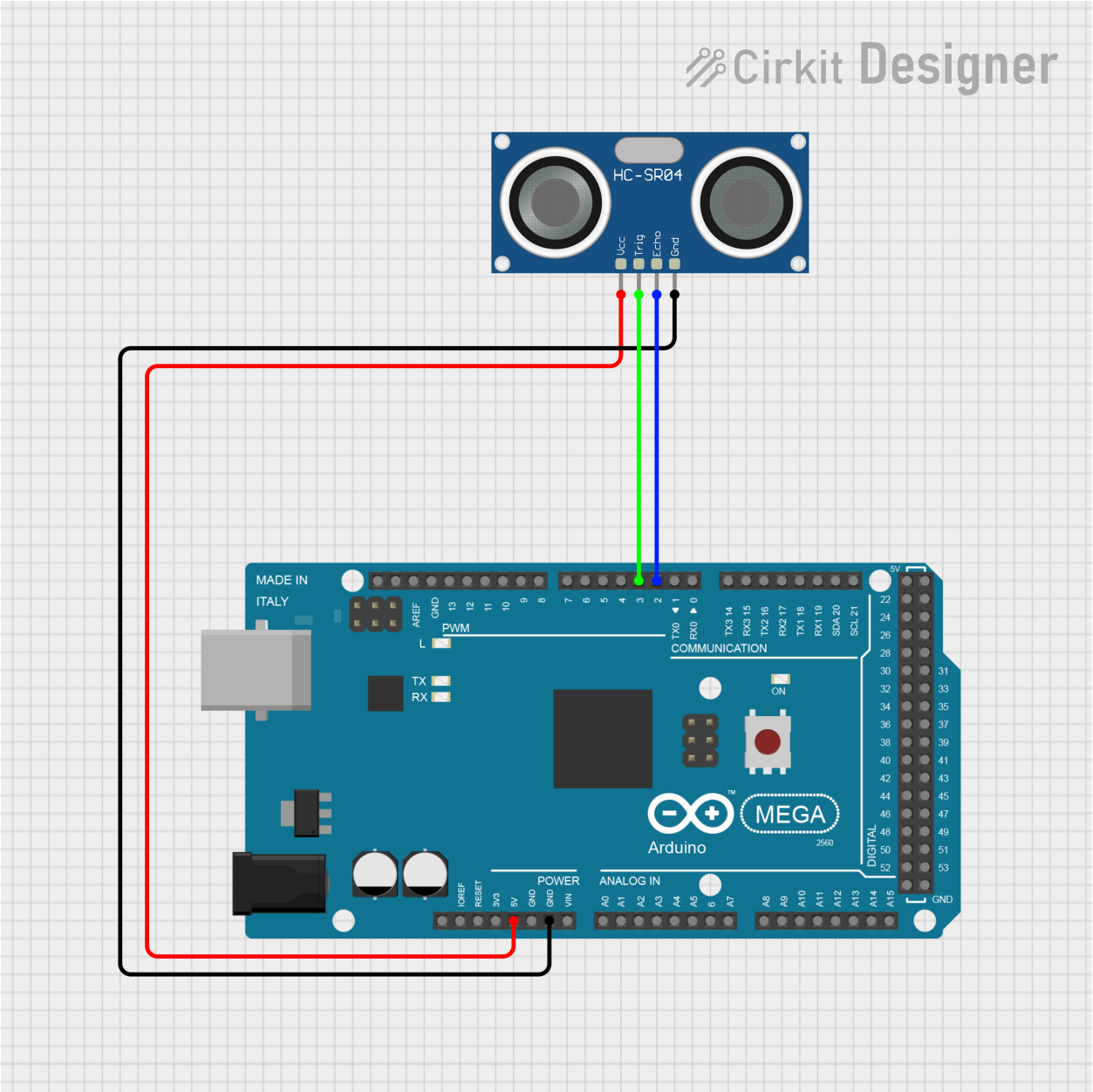
HC-SR04 pinout:



Controlling sensor using Arduino

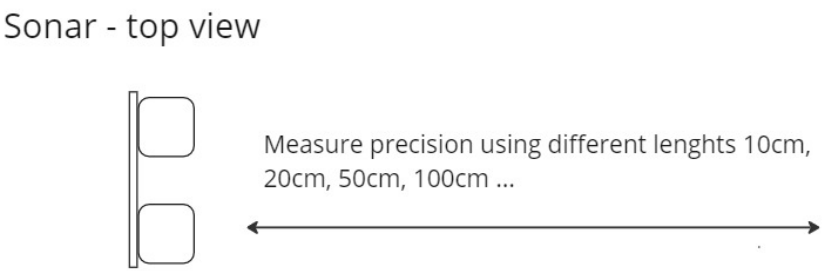
- Creating trigger pulse using digitalWrite() and delayMicroseconds()
- Reading length of the pulse with pulseIn()
- pulseIn() returns length of pulse in milliseconds, needs to be transformed into unit of length using speed of sound.

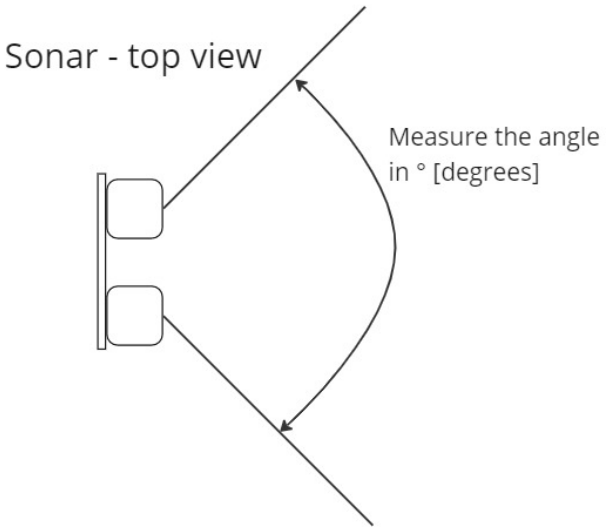
HC-SR04 - Scheme

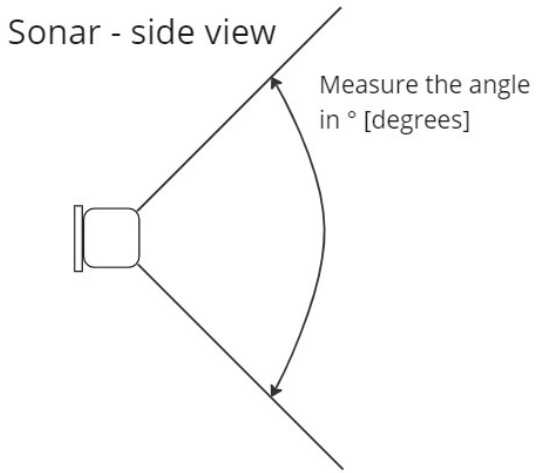


- Connect sensor to Arduino as it is in the scheme
- Open file HC-SR04_ranging.ino
- Write your own code for reading of the length of pulse
- Transform the length of pulse into centimeters
- Upload and test your work

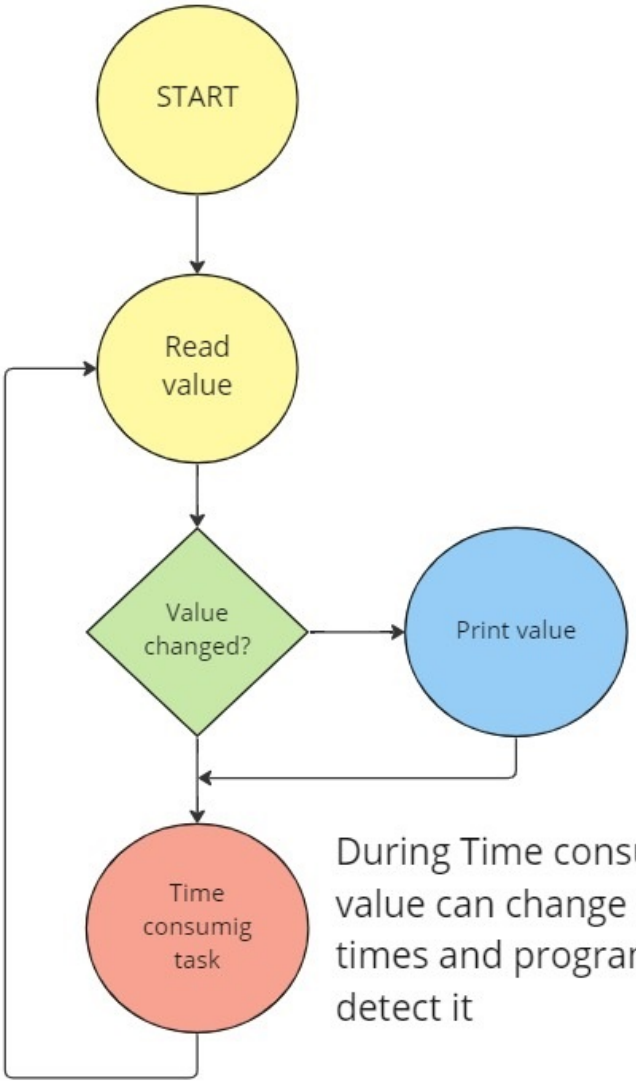
Measure the properties of the sensor. Also try different materials (Cloth, wood, plastic...) and see how sensor reacts.





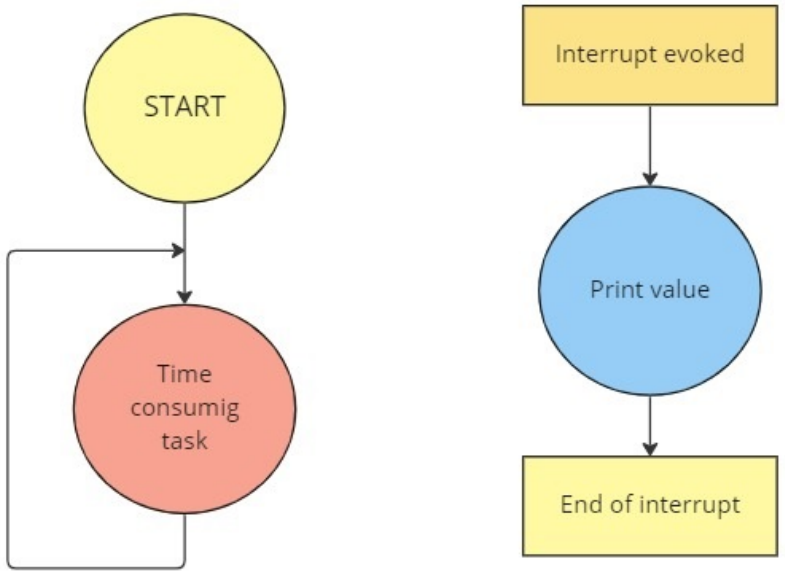


Active waiting



During Time consuming task value can change multiple times and program won't detect it

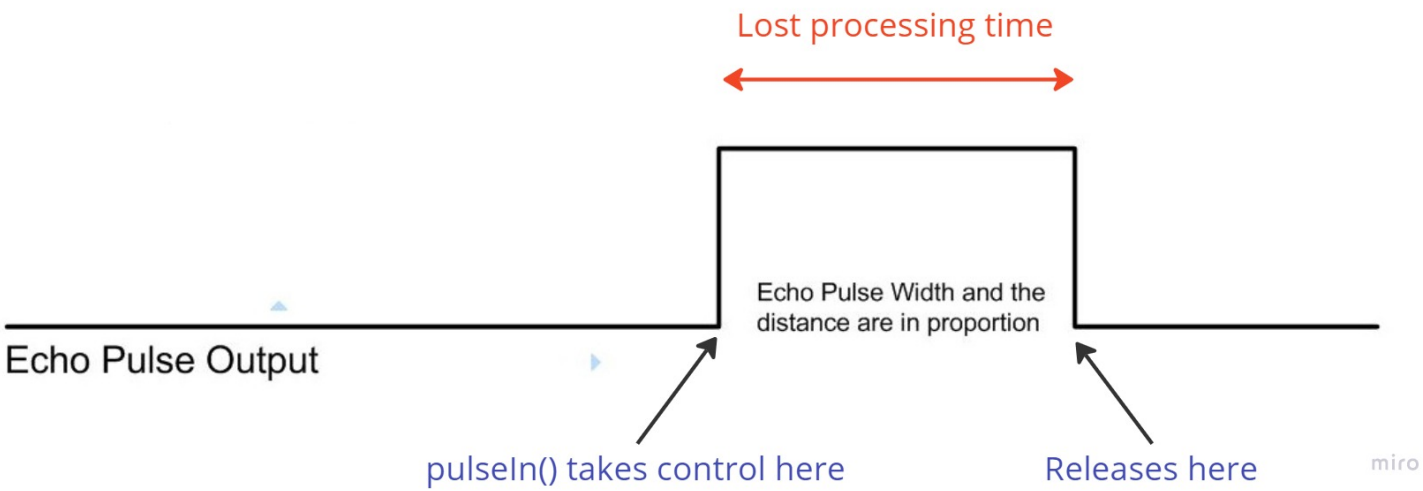
Interrupt



If value is changed:

- Interrupt is evoked
- Main process is paused
- Value is printed
- Interrupt is ended
- Main process is resumed

- pulseIn() uses active waiting for detecting the pulse
- Interrupts are only evoked when there is change



HC-SR04 - Better version

- Use the same scheme as in previous exercise.
- Open file HC-SR04_interrupt.ino
- Try and measure distance with code using interrupts. Is sensor behaving differently?

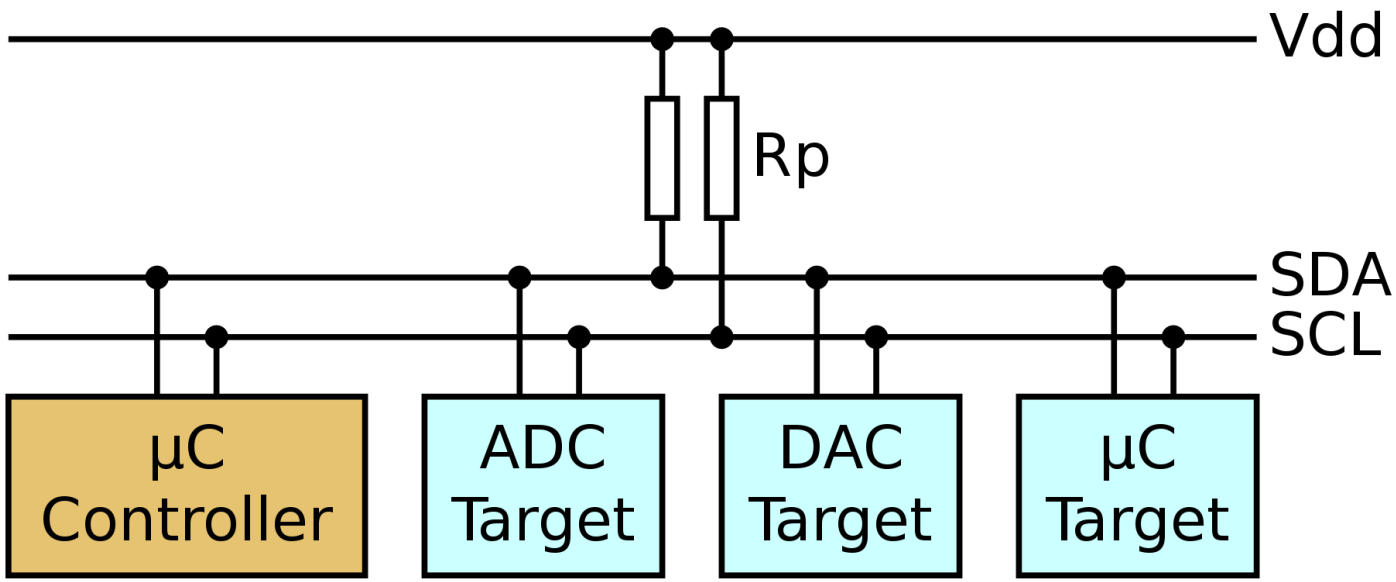
Convert distance sensor into speedometer.

- Open file HC-SR04_speedometer.ino
- Measure distance using any of the two previous codes
- Calculate derivation of the distance to get the speed
- Visualize the results using Serial plotter

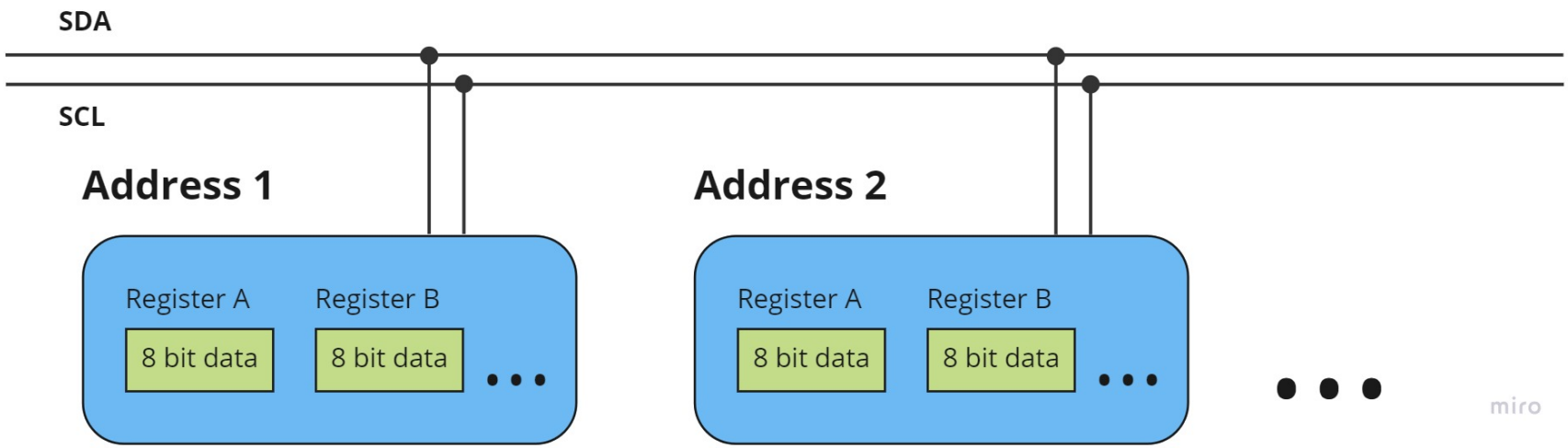
I2C Preview

- 2 wire bus - SDA, SCL
- SDA - data signal
- SCL - synchronizing clock signal
- Up to 127 devices
- Each device has unique address
- Master - Slave model

In order to work properly, there need to be pull-up resistor on both wires. Can you guess why?

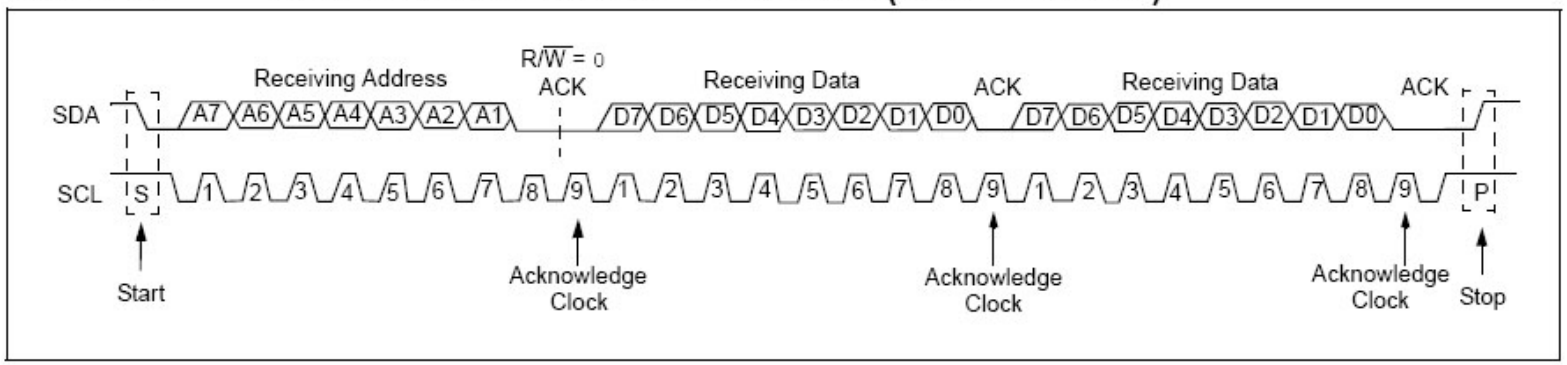


Each of the devices has its unique address. Inside each device are registers that have unique address but only inside of the device (there can be 2 registers with same address in different devices).



I2C uses 8-bit words for transmission of data. The first word is address, second is register location and then data itself follow.

FIGURE 1: TYPICAL I²C™ WRITE TRANSMISSION (7-BIT ADDRESS)



Arduino has for communication via I2C built in library wire.h.
Functions used for communication are pretty self-explanatory.

- beginTransmission() Begins I2C communication
- endTransmission() Ends I2C communication
- write() Writes byte to device
- requestFrom() Requests data from device
- available() Checks if requested data are available
- read() Reads one byte of available data
- write() Writes byte to device

Sample of a code that sends one byte from master to slave device.

```
Wire.beginTransmission(0x50);     // Start transmission with device with address 0x50

Wire.write(0x42);                 // select register 0x42
Wire.write(0x69);                 // write value 0x69 to register 0x42

Wire.endTransmission();           // Stop transmission
```

Sample of a code that reads one byte from slave device.
(Code runs at master device)

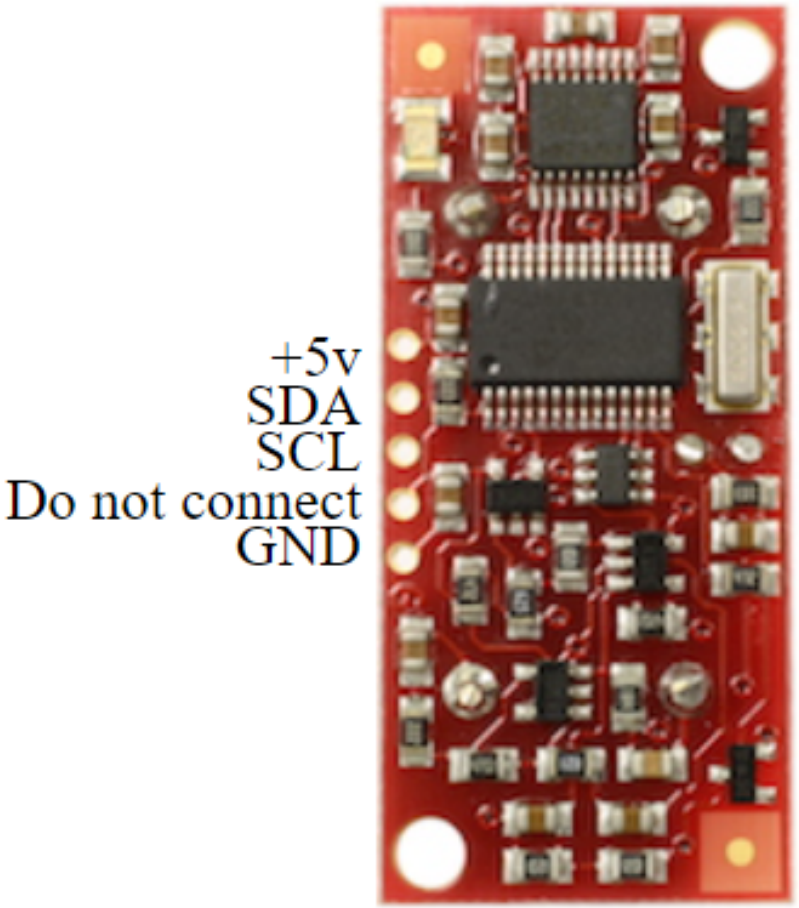
```
// Selecting from which register to read
Wire.beginTransmission(0x50); // Start transmission with device with address 0x50
Wire.write(0x42);             // select register 0x42
Wire.endTransmission();       // stop transmitting

Wire.requestFrom(0x50, 1);    // Request 1 byte from device with address 0x50


if (1 <= Wire.available()) {  // If byte was received
  value = Wire.read();        // Read received byte
}
```


SRF08

SRF08 - Difference between HC-SR04

- HC-SR04 costs 4€, while SRF08 costs 40€
- SRF08 uses I2C bus line while HC-SR04 uses trigger/echo
- SRF08 returns data in cm/inch/ms as HC-SR04 has only pulse length
- Can you guess how many wires it would take to connect 10x HC-SR04 and 10x SRF08?

Look at the pinout from the backside.



In order to communicate with SRF08 you need to know its unique address

- Address can be internally changed
- What would happen if there were 2 devices with same address?
- Full documentation can be found on `https://www.robot-electronics.co.uk/htm/srf08tech.html`

Registers needed to know when working with SRF08.

Register	Location	Read/Write
Command register	*0x00*	*Write*
Light Sensor	*0x01*	*Read*
Echo High Byte	*0x02*	*Read*
Echo Low Byte	*0x03*	*Read*

Here are commands that can be written to Command Register. Output unit is depended on which command is written to register.

Action	*Value*
Range - Result in inches	0x50
Range - Result in centimetes	0x51
Range - Result in miliseconds	0x52

SRF08 reading values bigger than 255

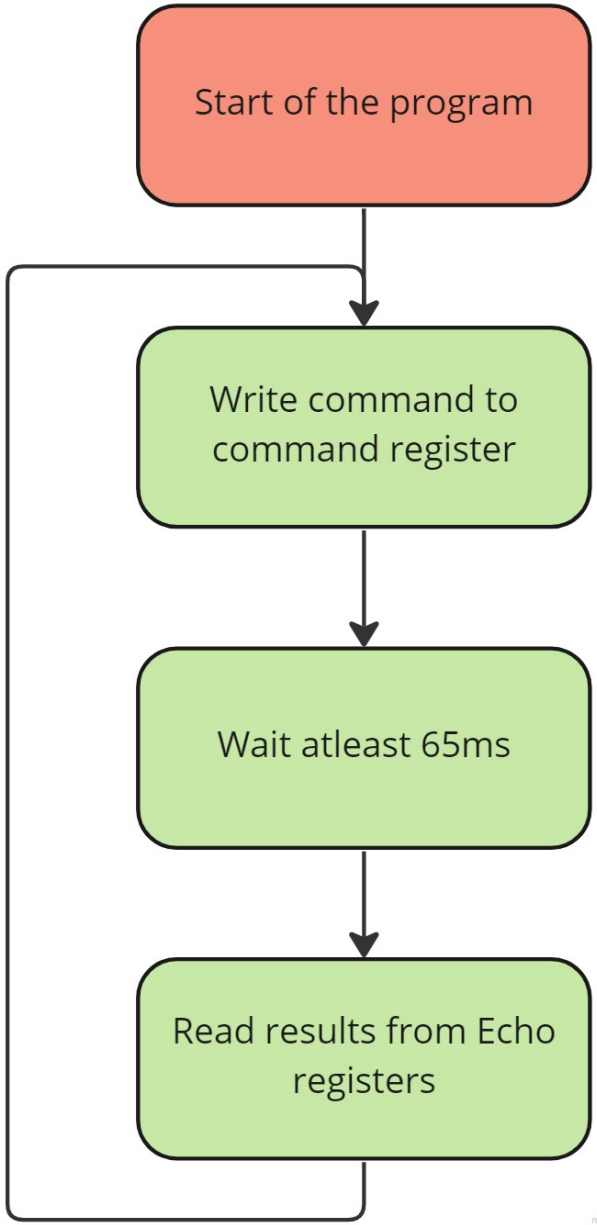
- Reading one byte gives range 0-255. What if we want to measure greater distance?
- Need to read 2 bytes Echo High Byte and Echo Low Byte
- Useful for reading result in milliseconds

Result	
Echo High Byte	Echo Low Byte
15 8	7 0

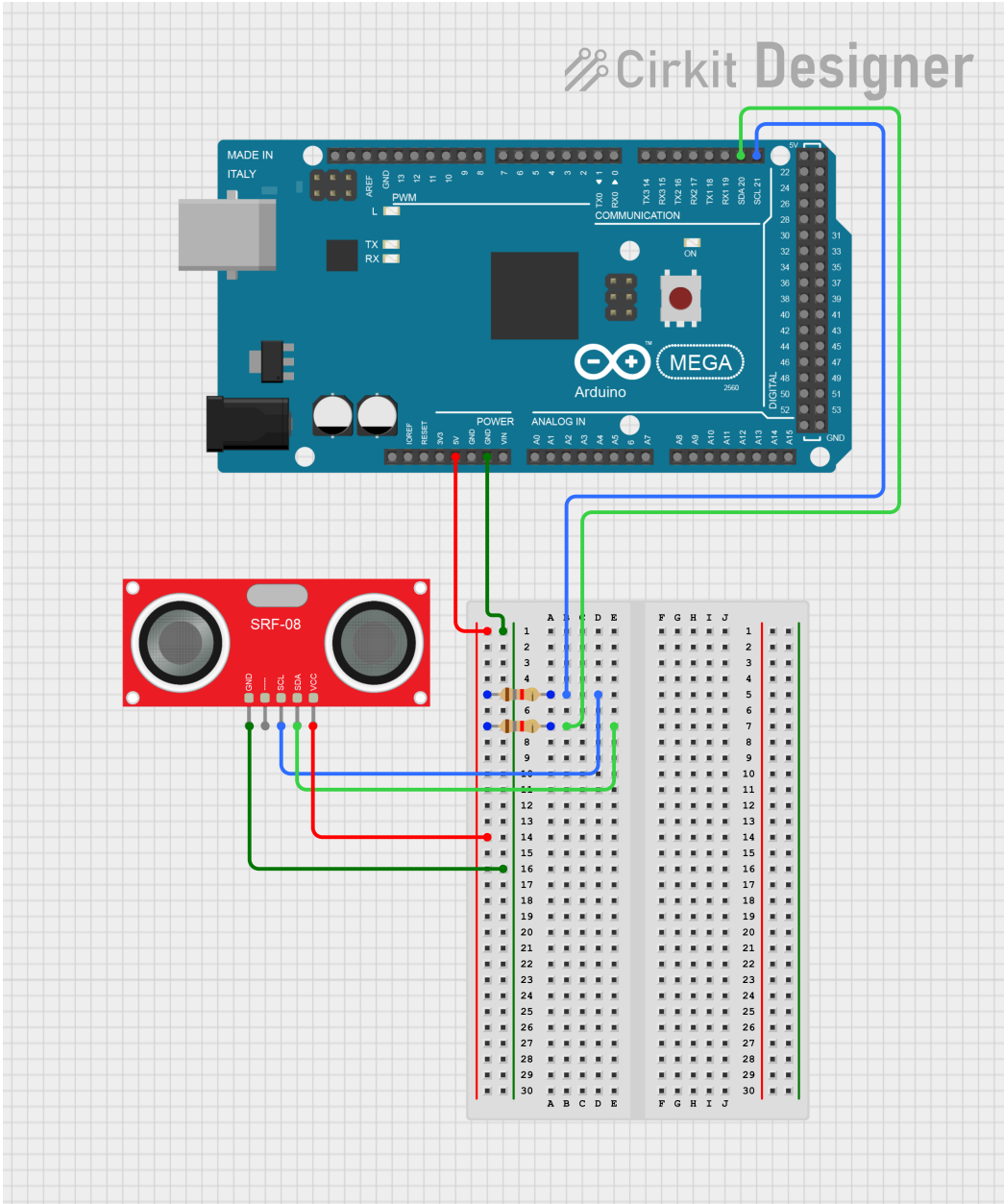
miro

Therefore Result = High Byte * 256 + Low Byte

SRF08 Ranging loop



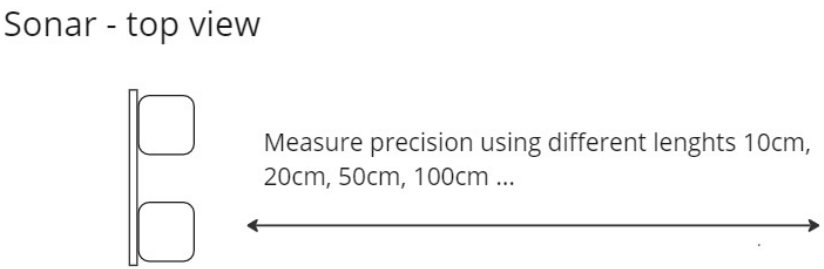
SRF08 Scheme

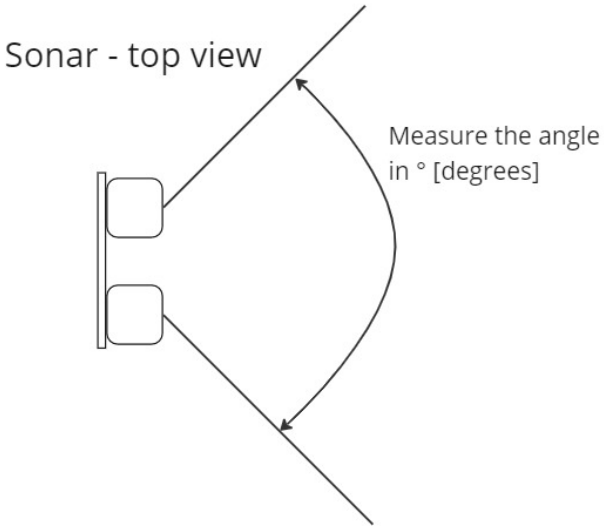


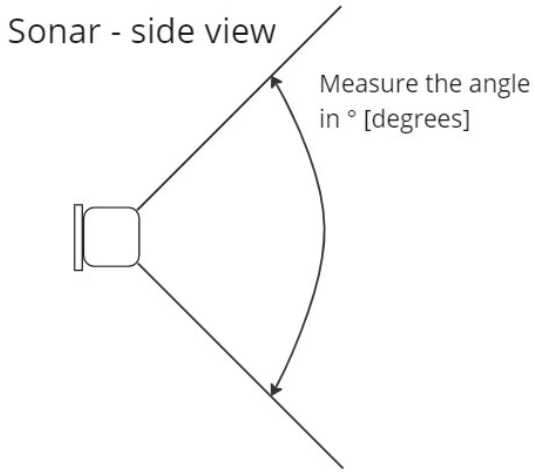
Use SRF08 to measure distance.

- Open file SRF-08_ranging.ino
- Send command to the command register to measure distance in centimeters
- Wait for at least 65 milliseconds
- Select register to read from
- Read from that register and visualize the result in Serial monitor/plotter
- Test your code by trying and measuring different distances

Test the sonar in same way as you did with the previous one - Measure the properties of the sensor. Also try different materials (Cloth, wood, plastic...) and see how sensor reacts.







As you might have noticed, this sensor is 2 in 1. Other than distance sensor, it contains light sensor.

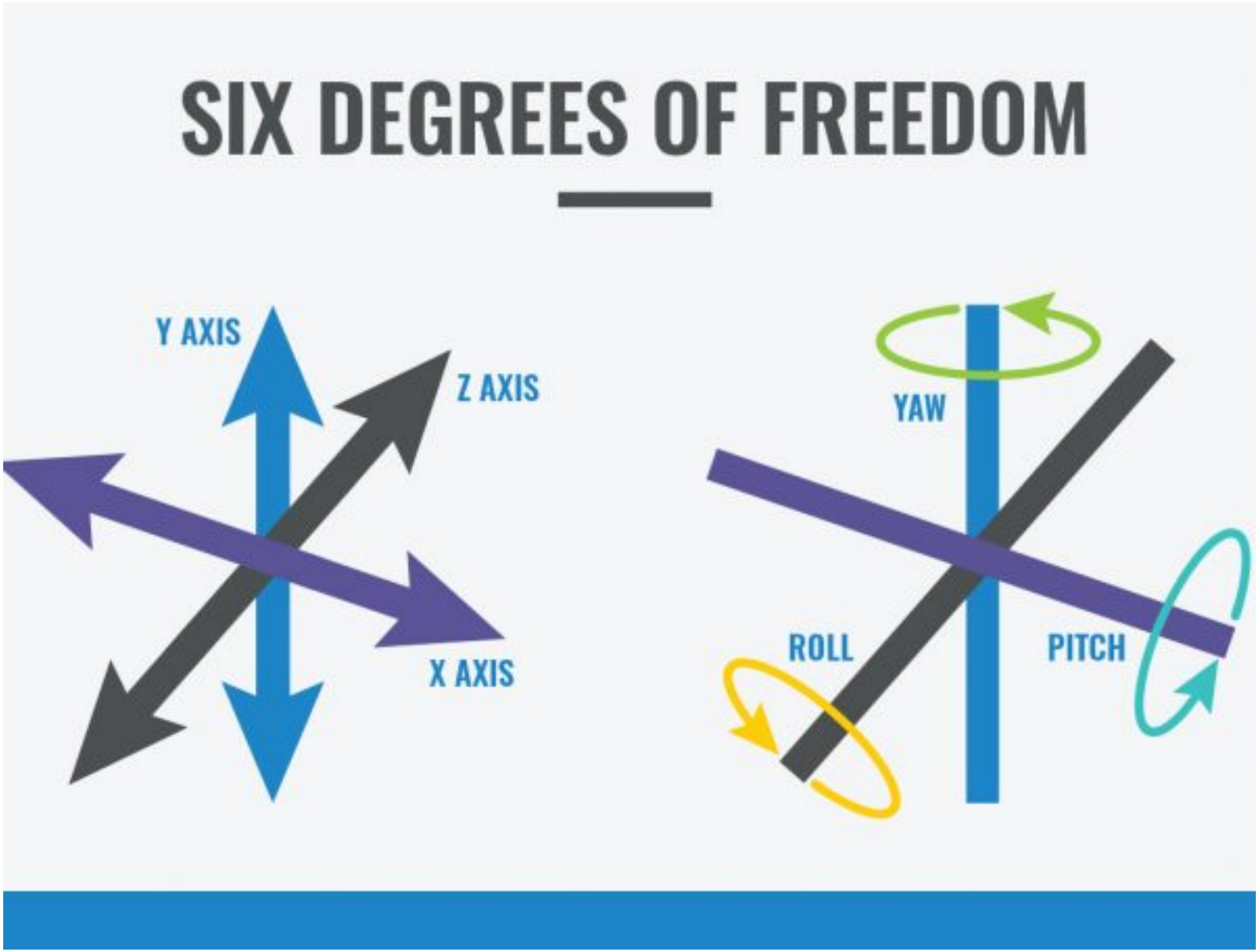
- Take a look at the code that reads data from distance sensor.
- Modify this code (or open SRF-08_light_sensor.ino) in a way, so that it also reads data from the light sensor and sends it to the PC using Serial.
- Test your code by pointing the light sensor to the source of light and away from it.

MinIMU - What is that

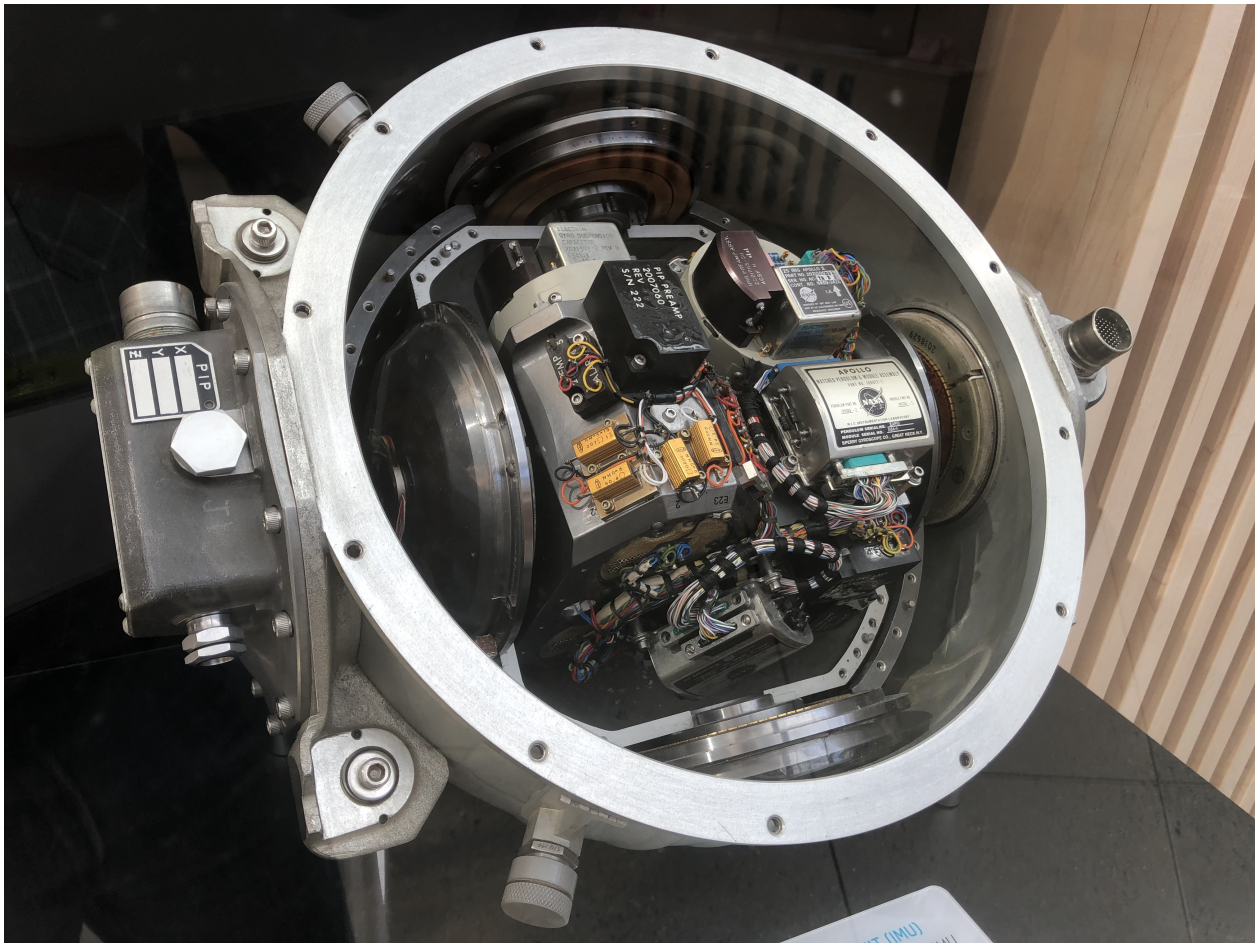
Inertial measurement unit is an electrical part that can detect forces like acceleration (even gravity). IMU used in this lab has (as per datasheet):

- Gyroscope
- Accelerometer
- Tap and double-tap sensing
- Activity / inactivity recognition
- Magnetometer
- Temperature sensor

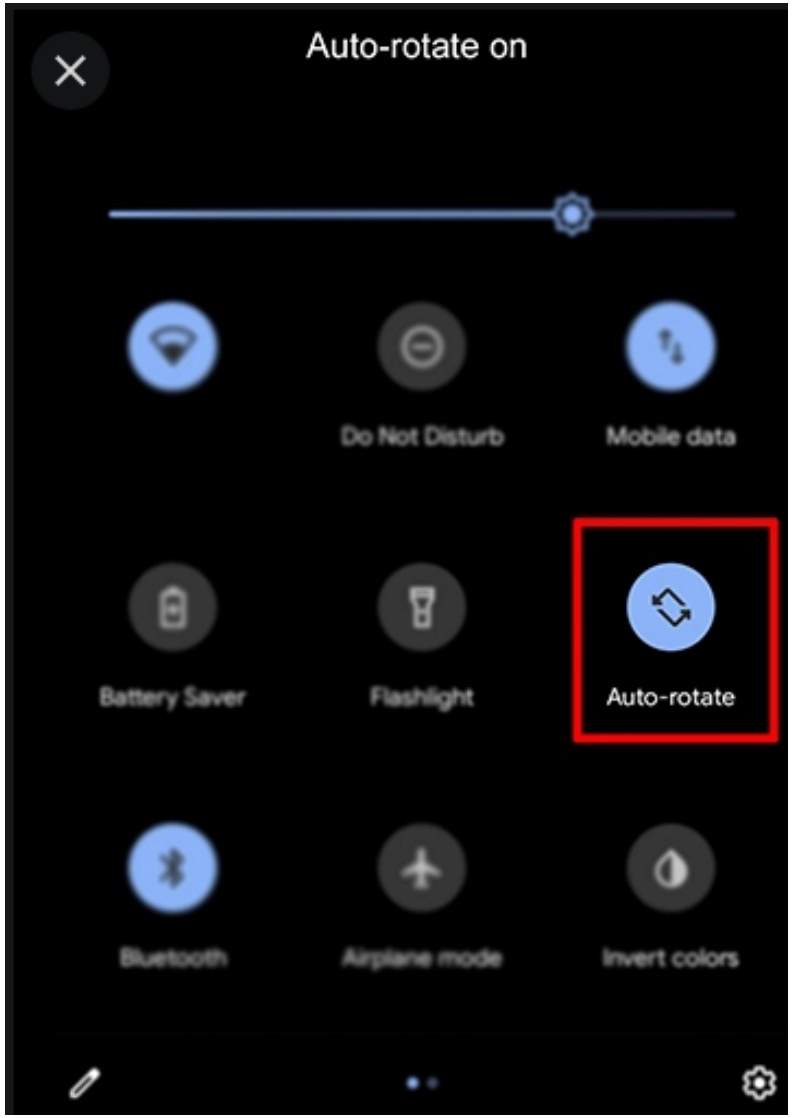
You can find whole documentation at
`https://www.pololu.com/product/2738`


SIX DEGREES OF FREEDOM
Y AXIS
Z AXIS
X AXIS
YAW
ROLL
PITCH

Spacecraft navigation



Every day in your life in your smartphones



MinIMU - How to communicate with it

- Communication is same as with SRF08, uses I2C
- 2 devices in one - Gyroscope/Accelerometer and Magnetometer
- Each has its own address

Gyroscope and accelerometer address	***0x6B***
Magnetometer address	***0x1E***

In order for the sensor to work correctly (or work at all) it needs to be initialized by writing data into control registers.

Control register	*Location*	*Read/Write*	*Desired value*
Linear acceleration	*0x10*	*Write*	*0x74*
Angular rate	*0x11*	*Write*	*0x7C*

Results of the reading are 16-bit, so for each value there need to be 2 readings

Angular Rate register	*Location*	*Read/Write*
X - axis low byte	*0x22*	*Read*
X - axis high byte	*0x23*	*Read*
Y - axis low byte	*0x24*	*Read*
Y - axis high byte	*0x25*	*Read*
Z - axis low byte	0x26	*Read*
Z - axis high byte	*0x27*	*Read*

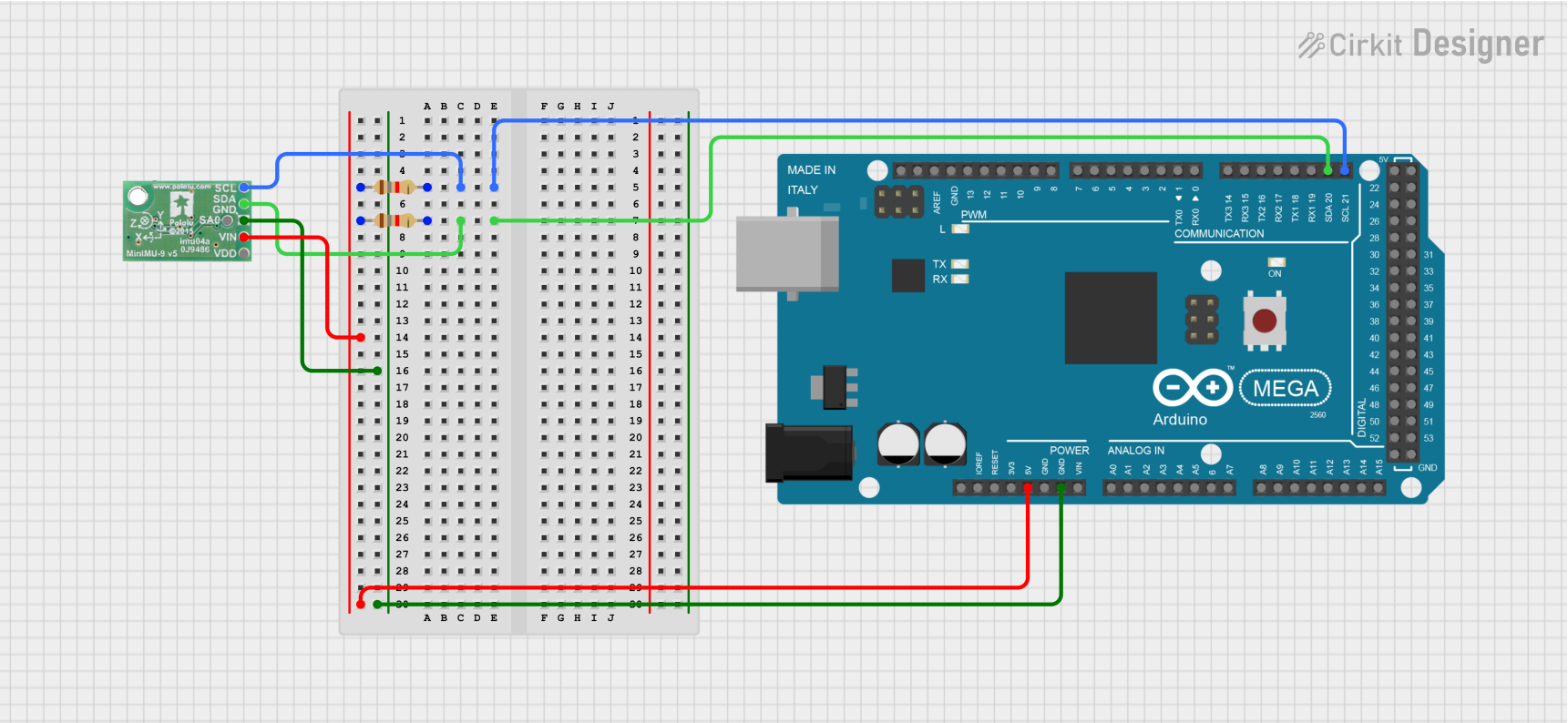
Results of the reading are 16-bit, so for each value there need to be 2 readings

Linear Acceleration register	*Location*	*Read/Write*
X - axis low byte	*0x28*	*Read*
X - axis high byte	*0x29*	*Read*
Y - axis low byte	*0x2A*	*Read*
Y - axis high byte	*0x2B*	*Read*
Z - axis low byte	0x2C	*Read*
Z - axis high byte	*0x2D*	*Read*

MinIMU - What do we even read

What does the read number from IMU mean?

- IMU detects only changes in speed and rotation.
- Result is 16-bit signed integer that corresponds to the selected range
- Linear acceleration unit is g, range is +- 16g
- Angular rate unit is dps (degrees per second), range is +- 2000dps


Cirkit Designer
SCL
SDA
GND
SA0
VIN
VDD
MinIMU-9 v5
MADE IN
ITALY
PWM
COMMUNICATION
TX
RX
ON
MEGA
Arduino
POWER
ANALOG IN
DIGITAL

Use MinIMU to measure linear acceleration

- Open file IMU_lin_acc.ino
- Send setup value to control register.
- Read data from IMU and combine High and Low byte together into one value
- Send the data into PC using Serial communication
- Test your implementation by moving IMU

IMU is showing acceleration even when you don't move with it. Why is that?

Convert the 16-bit number that you read into unit of g.

- You need to find a constant with which to divide the value
- Experiment with IMU and try to measure gravitational acceleration and to which value 1g corresponds

Use MinIMU to measure angular rate (in its essence the same as linear acceleration)

- Open file IMU_ang_rate.ino
- Send setup value to control register.
- Read data from IMU and combine High and Low byte together into one value
- Send the data into PC using Serial communication
- Test your implementation by rotating IMU

Get the relative angle of IMU inclination. Integrate the values (Calculate the sum of the values)

- Continue modifying your file or open file IMU_ang_rate_integral.ino
- Follow the instructions and implement the integration.
- Send the data into PC using Serial communication
- Test your implementation by rotating IMU

This MinIMU also offers temperature sensor. It does not have any setup registers.

Temperature register	*Location*	*Read/Write*
Output low byte	*0x20*	*Read*
Output high byte	*0x21*	*Read*

- Open file IMU_temperature.ino
- Read data from IMU and combine High and Low byte together into one value
- Send the data into PC using Serial communication

Try to warm up and cool down the IMU (for example by touching it)

- Magnetometer measures magnetic fields
- 3 axis
- Can measure Earth's magnetic field
- Can be used as compass

In order for the magnetometer to work correctly (or work at all) it needs to be initialized by writing data into control registers.

Control register	*Location*	*Read/Write*	*Desired value*
Magnetometer	*0x22*	*Write*	*0x00*

Results of the reading are 16-bit, so for each value there need to be 2 readings

Magnetometer register	Location	Read/Write
X - axis low byte	0x28	Read
X - axis high byte	0x29	Read
Y - axis low byte	0x2A	Read
Y - axis high byte	0x2B	Read
Z - axis low byte	0x2C	Read
Z - axis high byte	0x2D	Read

- Open file IMU_magnetometer.ino
- Send setup value to control register.
- Read data from IMU and combine High and Low byte together into one value
- Send the data into PC using Serial communication
- Test your implementation by rotating IMU
- Put to proximity of IMU metallic object. What do you observe?

Your objective: Find NORTH

Thank You For Your Attention !