

# 1 Laboratory 1

Welcome to the third laboratory. Goal of this lab is to get you know with basics sensors and and their way of interacting with real world.

## 2 Prerequisites

It is expected that you have knowledge from previous laboratories (prototyping electrical circuits using breadboard).

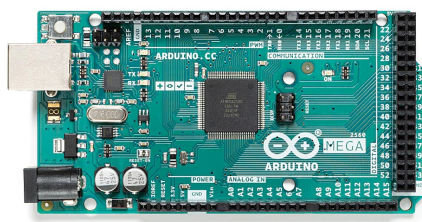
## 3 Goals of the laboratory

Goals of this laboratory are set to introduce you to basics of sensors. You will learn:

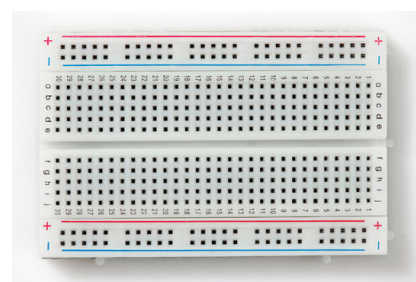
- Measuring distance using HC-SR04 ultrasonic distance sensor
- I2C protocol
- Measuring distance using SRF08 ultrasonic distance sensor
- Work with IMU (Inertial Measurement Unit) and reading its data

## 4 Hardware used in this laboratory

Here is list of hardware and electrical parts used in this laboratory



(a) Arduino Mega



(b) Breadboard

Figure 1: Core of the laboratory



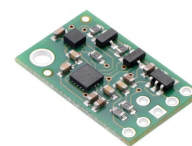
(a) Resistor



(b) HC-SR04



(c) SRF08



(d) MinIMU-9 v5

Figure 2: Electrical parts

**Arduino Mega** Contains XX processor. Your code will run on it

**Breadboard** Used for creating electronics circuits

**Resistor** Used for limiting current for electronics parts (like LED)

**HC-SR04** Cheap ultrasonic distance sensor

**SRF08** Pricy ultrasonic distance sensor

**MinIMU-9 v5** Inertial measurement unit

## 5 Theory

### 5.1 Sensors

Sensors are electrical devices that are able to transform physical phenomenon to electrical output signal in order for the electrical devices to interact with real world. Different sensors detect different phenomena like light, sound, speed, force, rotation, etc.

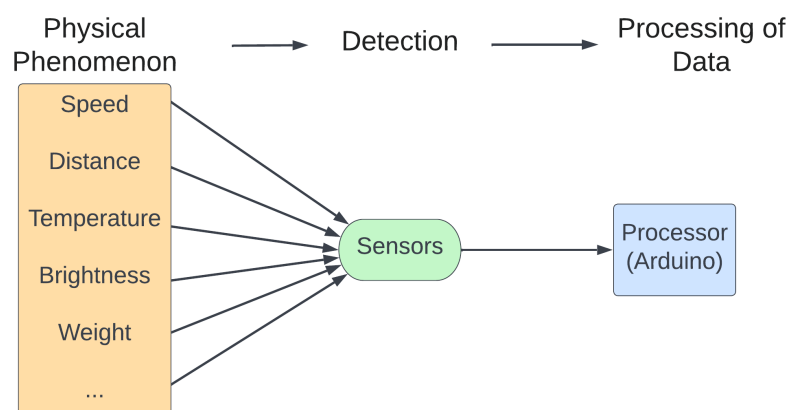


Figure 3: How sensors work

### 5.2 HC-SR04

HC-SR04 is ultrasonic distance sensor that uses (40kHz) ultrasound waves to detect distance of the objects from this sensor. It has 2 parts Speaker which creates this soundwaves and microphone that detects reflected waves. From this basic principle it is possible to measure time it took sound to travel and from that calculate the actual distance.

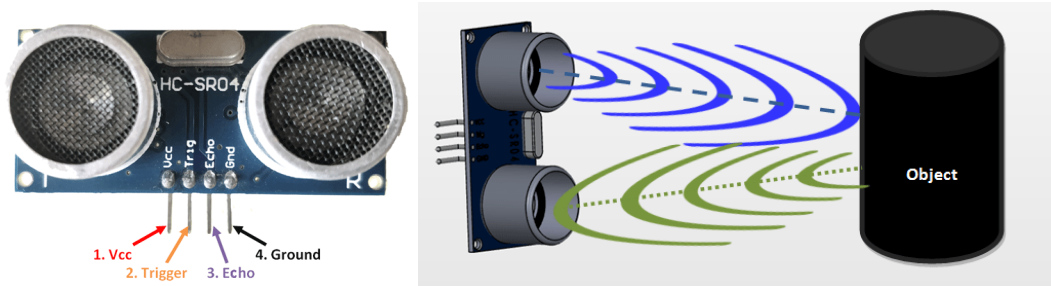


Figure 4: HC-SR04 and its pinout

Operating this sensor is relatively simple. It has 2 communication pins - *Trigger* and *Echo*. In order to take measurement there needs to be 10 microseconds pulse on Trigger pin. Then sensor fires its sound waves and length of pulse on the Echo pin is proportional to the distance measured (longer distance = longer pulse). Figure XY shows this as diagram.

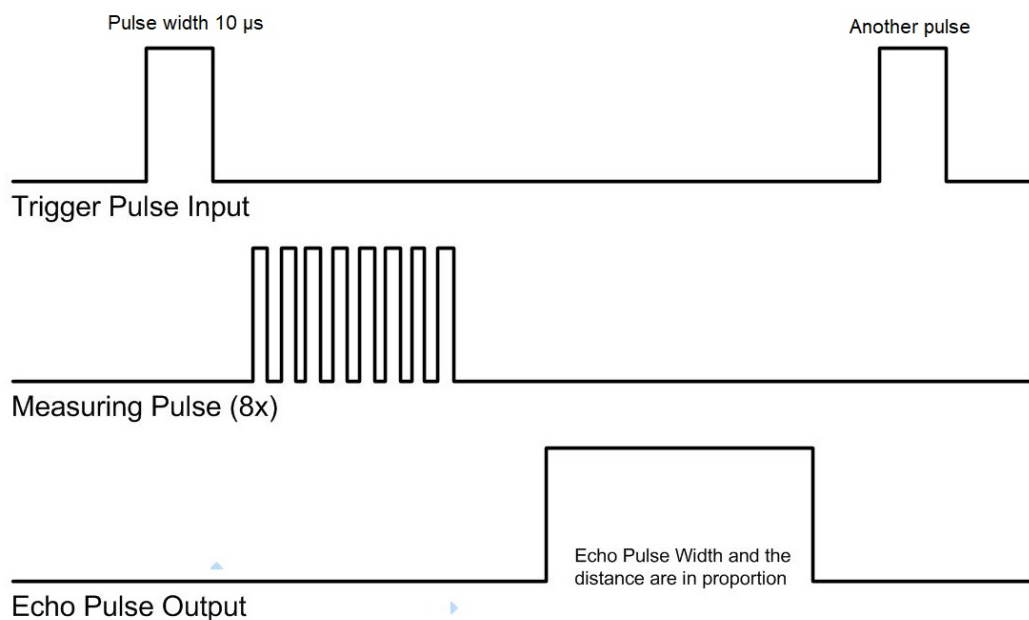


Figure 5: How sensors work

How to use arduino with this sensor. For creating Trigger pulse input can be used functions `digitalWrite(pin, HIGH/LOW)` and `delayMicroseconds(microseconds)`. For reading Echo pulse there is built in function `pulseIn(pin,HIGH)` which returns length of the pulse in microseconds.

In order to get the measurement of distance in centimeters you need to calculate it using speed of sound which is 343 ms-1 or 0.034cm us-1 Also Echo pulse length is equal to *two way* trip of the sound. Therefore the pulse length needs to be divided by 2.

$$Distance = Echo\_pulse\_length * 0.034/2$$

### 5.3 Interrupts

Interrupts are special way of handling events that need immediate attention. In order to be able to handle interrupts processor needs to have hardware support for interrupts. When interrupt occurs - for

example a change of value on a pin, processor stops its main process and starts its interrupt service routine. After the routine is finished processor returns to its main process.

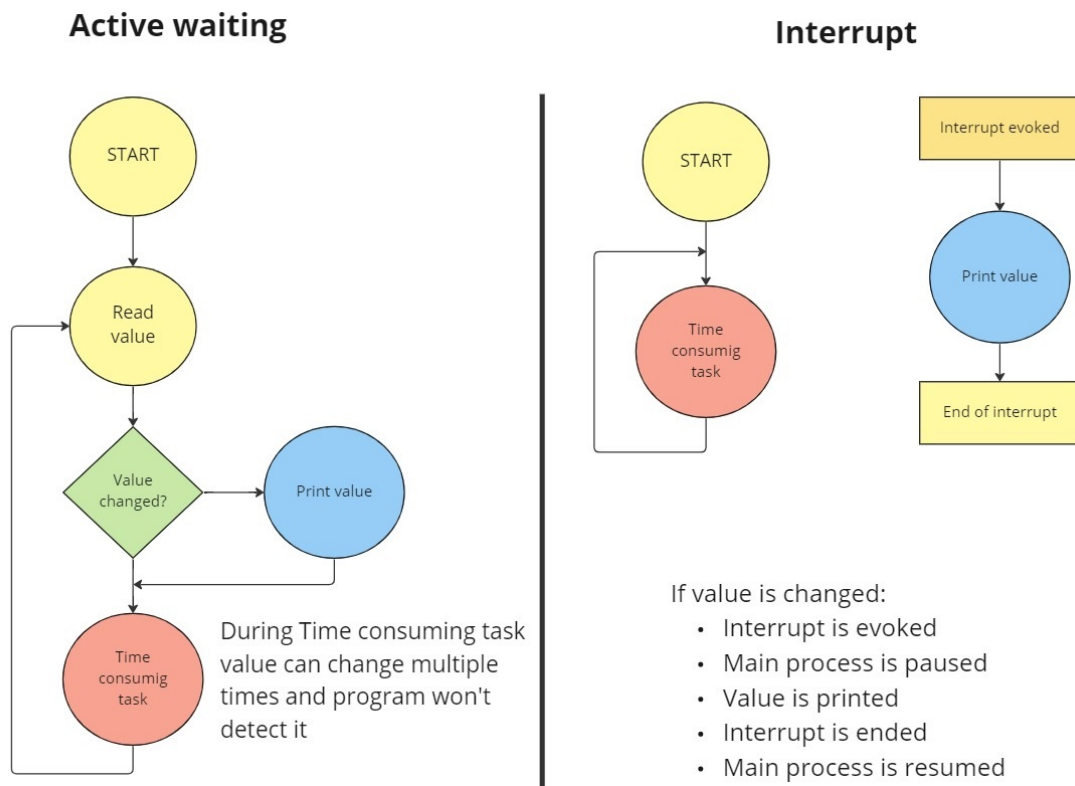


Figure 6: How sensors work

## 5.4 I2C

### 5.4.1 Basics

I2C is synchronous bidirectional bus with architecture master-slave. It has 2 lines SDA and SCL where SDA is data and SCL is clock signal. I2C uses open collector which means that on each wire there has to be pull up resistor 2kohm (some devices might have those resistors built in).

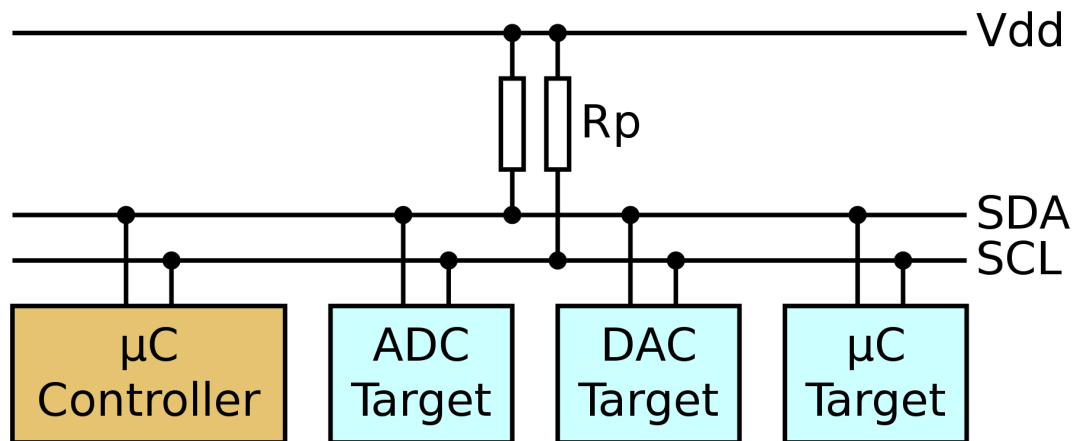


Figure 7: How sensors work

Each device on the bus has its own 7-bit address which means that on one I2C bus can be up to 127 devices. Each device has its own internal registers, to which master can write data or request data from it. Each register has its own unique address, but only inside of the device (there can be 2 registers with same address in different devices).

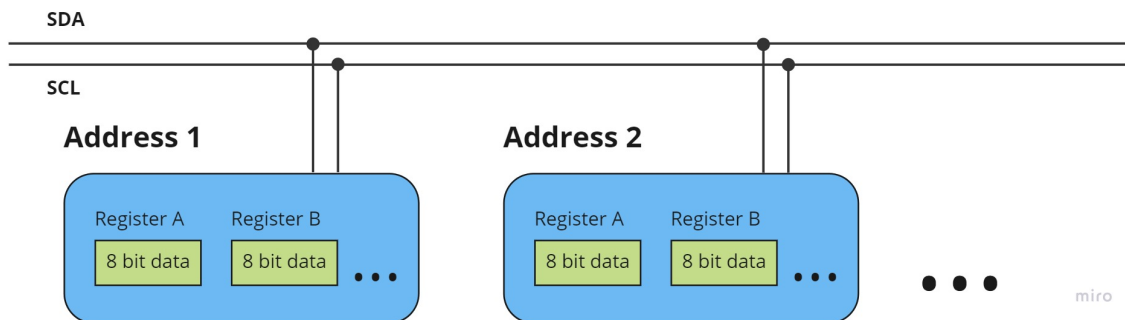


Figure 8: How sensors work

I2C uses 8 bit protocol - meaning that length of one word is 8 bit. After each message there is acknowledge bit. First word is address and the rest is data.

FIGURE 1: TYPICAL I<sup>2</sup>C™ WRITE TRANSMISSION (7-BIT ADDRESS)

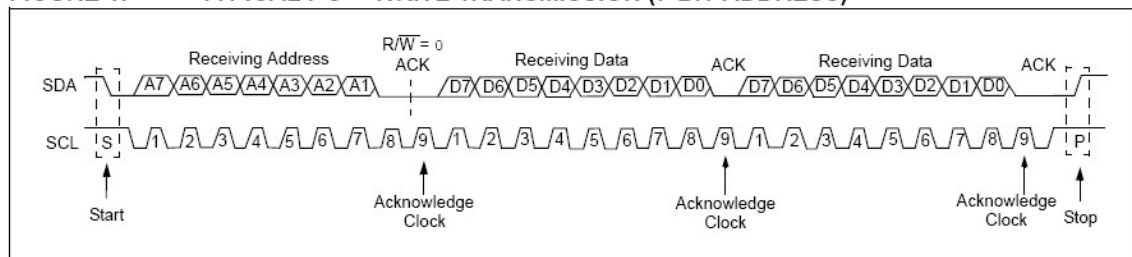


Figure 9: How sensors work

### 5.4.2 Arduino preview

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

figure XY shows samples of writing data as master device from slave device.

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

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

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

Figure 10: How sensors work

figure XY shows samples of reading data as master device from slave device.

```
// Selecting from which register to read
Wire.beginTransmission(0x50); // Start transmission with device with address 0x50
Wire.write(byte(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
}
```

Figure 11: How sensors work

## 5.5 SRF08

SRF08, same as HC-SR04, is ultrasonic distance sensor. Working principle is the same, sonar fires a sound wave and measures time that it takes for the sound wave to reflect back into its microphone.



Figure 12: How sensors work

Where is the difference between HC-SR04 and SRF08? While HC-SR04 uses pins trigger and echo, SRF08 is more sophisticated and uses I2C protocol for communication. Another difference is in cost. HC-SR04 costs 4\$ and SRF08 costs 40\$.

I2C specifications of SRF08 are as following. Like every I2C device it has its own address. This address can be internally changed.

Address	0x70
---------	------

Figure 13: How sensors work

TODO emph na vsetky nazvy a tak

Internal registers of SRF08 are place where all necessary data needed to operated the distance sensor are stored. Firstly there is Command register. This register is used for starting ranging session. Session is started by writing command value into this register.

Light sensor register is used for reading output data from light sensor that is present at SRF08.

Echo Low byte and Echo High byte registers are output registers from the measuring distance itself. Measure distance can be in cm inch or milliseconds (as can be seen on figure XY). In order to read values bigger than 255, the value is splitted into 2 registers. this is explained in XY

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

Figure 14: How sensors work

Reading data from SRF08 goes as following. Firstly there needs to be written value into command register that will initiate the ranging (values are explained on figure 69). Then the sensor starts its ranging sequence - for the next 65ms it will not respond to any commands, as it is performing the ranging. After this time is passed, result is ready in Echo registers.

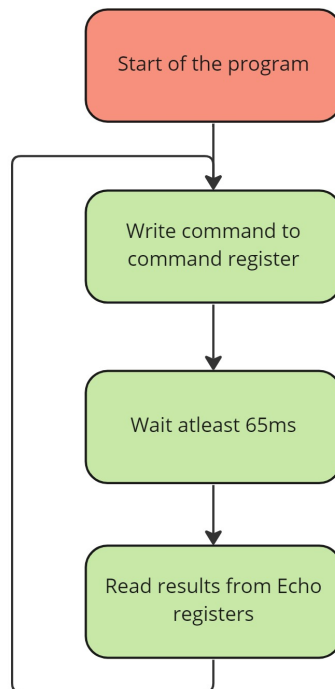


Figure 15: How sensors work

Behaviour of sensor depends on value written into its command register. There are 3 different values - an each specifies what will be the output unit of result. It can be centimeter, inch, or milliseconds.

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

Figure 16: How sensors work

Output registers are 8 bit so that their values can be send using I2C communication. That means that in one register can be value in range <0,255>. But what happens when distance measured is higher than 2.55 meters? This is why result is stored in 2 8-bit registers, which gives 16 bit range <0,65536>. In order to get this extended value range, values need to be combined in master device that requests the data. High byte needs to be shifted by 8 bits to the left (multiplied by 256) and Low byte can stay as it is.



Result					
Echo High Byte			Echo Low Byte		
15	....	8	7	....	0

Figure 17: How sensors work

$$Result = HighByte * 256 + Lowbyte$$

## 5.6 IMU

Another sensor used in this lab is IMU - Inertial Measurement Unit. Its is combination of accelerometers, gyroscopes and magnetometers. Purpose of IMU is to measure objects specific force.

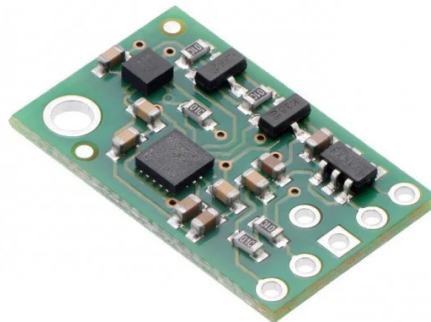


Figure 18: How sensors work

IMU used in this lab is MinIMU-9 v5. Its small packaging contains 2 modules - Gyroscope and accelerometer. MinIMU-9 v5 uses for communication I2C bus. Each of the modules has its own I2C address. First module is LSM6DS33 - gyroscope and accelerometer. Second is LIS3MDL - magnetometer. Each of the modules also offers temperature sensor.

Gyroscope address	0x6B
Magnetometer address	0x1E

Figure 19: How sensors work

### 5.6.1 Gyroscope

Gyroscopes goal is to measure forces objects specific forces. In this case gyroscope consists of 2 sensors - one for linear acceleration and other one for angular rate. This gives the gyroscope to measure objects specific forces along 6 degrees freedom.

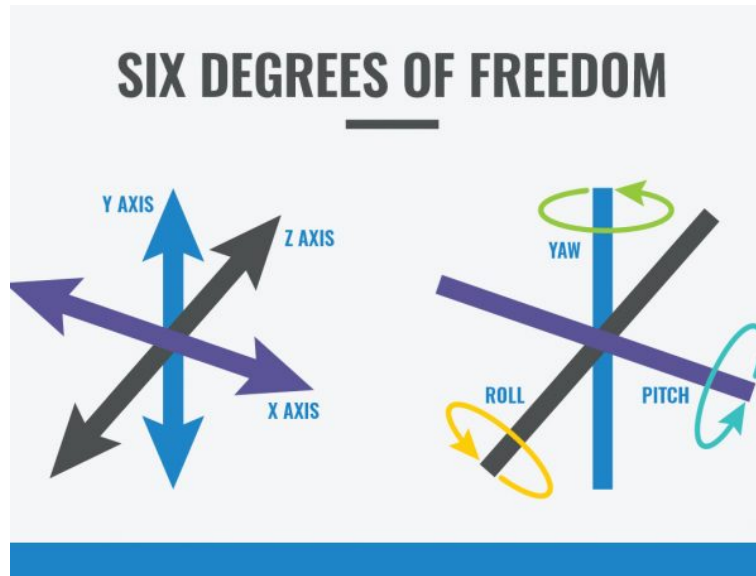


Figure 20: How sensors work

Gyroscope can measure linear acceleration of object along 3 axes X, Y, Z and its angular rate along this axes, called Pitch, Yaw, Roll.

Communication with each part of the gyroscope goes as following: Firstly there needs to be written setup value into control register. This value says what should be the frequency of sampling and its given range of operation. After value is set gyroscope will perform measurement as per given frequency and other devices can read its data from its output registers.

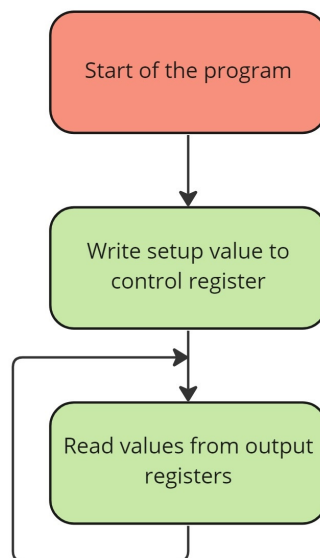


Figure 21: How sensors work

Use this values as setup values. It sets up the frequency of sampling to 833Hz, gives range for linear acceleration  $\pm 2g$  and gives range for angular rate of 2000dps.

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

Figure 22: How sensors work

### 5.6.2 Gyroscope - Linear acceleration

Linear acceleration sensor measures change in speed of the object along its axes. Output is along 3 axes X, Y, Z and each output is 16 bit signed integer. As I2C supports only sending 8bit numbers, the output value is split among 2 8 bit registers. Way of combining the numbers is explained at Figure XY. Output range is in that way  $\langle 32768, 32767 \rangle$  which corresponds to range that was selected in setup register, in case of this laboratory it is  $\langle -16g, +16g \rangle$ . Unit of g is way to measure acceleration an 1g is equal to  $9.81ms^{-1}$ .

Measuring loop of linear acceleration is explained at figure XY. Setup register and its value for accelerometer is described at figure XY and output registers are at Figure XY.

<b>Linear Acceleration register</b>	<b>Location</b>	<b>Read/Write</b>
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

Figure 23: How sensors work

### 5.6.3 Gyroscope - Angular rate

Angular rate sensor measure changes in rotation of the object along its axes. Output is along 3 axes X,Y,Z and rotation along this axes is called Pitch, Yaw, Roll. As I2C supports only sending

8bit numbers, the output value is split among 2 8 bit registers. Way of combining the numbers is explained at Figure XY. Output range is in that way <32768, 32767> which corresponds to range that was selected in setup register, in case of this laboratory it is <-2000dps, 2000dps>. Unit of dps means degrees per seconds.

Measuring loop of linear acceleration is explained at figure XY. Setup register and its value for accelerometer is described at figure XY and output registers are at Figure XY.

<b>Angular Rate register</b>	<b>Location</b>	<b>Read/Write</b>
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

Figure 24: How sensors work

#### 5.6.4 Gyroscope - Temperature sensor

Another extra sensor that gyroscope offers is temperature sensor. There is not much info about this sensor even in documentation. Its output is one 16 bit signed integer and most importantly it is not calibrated - meaning that it has no output unit and its up to user to find out the ranges and calibrate the sensor

Measuring loop is the same and with accelerometer and linear acceleration. Firstly there needs to be command written into linear acceleration register (to start up the component). After that result can be read from output registers.

<b>Temperature register</b>	<b>Location</b>	<b>Read/Write</b>
Output low byte	0x20	Read
Output high byte	0x21	Read

Figure 25: How sensors work

### 5.6.5 Magnetometer

Another module that MinIMU-9 v5 offers is magnetometer. Magnetometer is device that measures magnetic field, in this case in 3 axes - X, Y, Z. Similarly as gyroscope uses I2C for communication. It has its own address 0xAA (as it is another device on I2C bus) as could be seen at figure XY. It has its own command register that needs to be set up before reading from output registers.

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

Figure 26: How sensors work

Output number is 16-bit number, but as I2C supports only sending 8bit numbers, the output value is split among 2 8 bit registers. Way of combining the numbers is explained at Figure XY. Output range is in that way <32768, 32767> which corresponds to range that was selected in setup register, in case of this laboratory it is <-2G, 2G>. Unit of G is meant for Gauss, which is unit for measuring magnetic flux. 10 000 Gauss = 1 Tesla.

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

Figure 27: How sensors work

Warning!!! Do not put strong magnets into proximity of magnetometer

## 6 Exercises


IDEAS: spravit cheat sheet kde by boli vsetky funkcie ktore sa postupne беру (ako napr pin mode a tak)

TODO: nazvy suborov, cesty pomocou bold

TODO: spravit prehlad toho co sa bude brat

### 6.1 HC-SR04 - Ranging

Measure distance using HC-SR04 sensor.

1. Open file **HC-SR04\_ranging.ino** with Arduino IDE using **File -> Open...**
2. Take a look at **HC-SR04\_scheme.png** and create circuit as it is shown on image
3. Follow the instructions inside and write your own code that reads length of pulse and converts milliseconds into centimeters
4. Upload your code by clicking on Upload button  or pressing Ctrl+U
5. Test the sensor by pointing it to different sides and measuring different distances

### 6.2 HC-SR04 - Testing the sensor

Test the abilities and range of the sensor.

1. Take a look at Figure 3

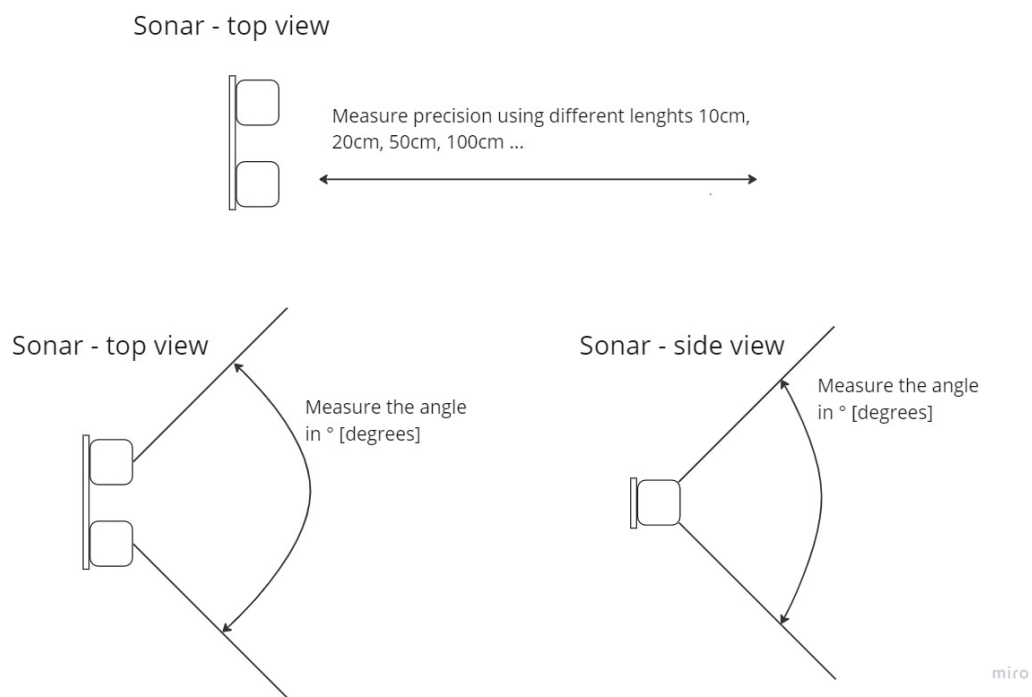



Figure 28: hahaha

2. Test precision of the sensor by placing object in known distance from sensor and observing output
3. Find out what is the angle/cone of sensors detection.
4. Try detection with different materials. Do all materials behave the same?
5. Discuss your results


### 6.3 HC-SR04 - Interrupt method

Previous implementation of ranging uses active waiting - meaning that processor cant be used for anything else while it performs measurement. Take a look at this implementation that uses interrupts.

1. Open file **HC-SR04\_interrupt.ino** with Arduino IDE using **File -> Open...**
2. Take a look at the implementation.
3. Upload your code by clicking on Upload button  or pressing Ctrl+U
4. Test the sensor and see if you can spot difference in behaviour compared to previous method.


### 6.4 HC-SR04 - Measuring speed

Test the accuracy and

1. Open file **HC-SR04\_speedmeter.ino** with Arduino IDE using **File -> Open...**
2. Follow the instructions inside and write your own code that calculates the speed of moving object in front of the sensor
3. Upload your code by clicking on Upload button  or pressing Ctrl+U
4. Test your code by moving object in front of the sensor

### 6.5 SRF08 - ranging

sem dat aj centimetre aj miliseconds - skladanie registrov Measure distance using SRF08 sensor.

1. Open file **SRF08\_ranging.ino** with Arduino IDE using **File -> Open...**
2. Take a look at **SRF08\_scheme.png** and create circuit as it is shown on image
3. Implement the control loop by sending command to sensor, waiting at least 65 ms and then reading the result from output register
4. Look at the results using Serial monitor and test your implementation by pointing sensor to different sides
5. Bonus exercise: Implement measuring  or pressing Ctrl+U
6. Test the sensor by pointing it to different sides and measuring different distances

## 6.6 SRF08 - Light sensor

SRF08 also offers lights sensor. Write a code that reads data from light sensor.

1. Open file **SRF08\_light\_sensor.ino**
2. Follow the instructions inside and write your own code that reads data from the light sensor and sends it to PC via Serial communication
3. Upload and test your implementation by pointing the light sensor towards light source and away from it.

## 6.7 MinIMU - measuring linear acceleration

Use MinIMU for measuring linear acceleration.

1. Open file **IMU\_lin\_acc.ino**
2. Follow the instructions inside and write your own code that sets up the IMU by writing setup value into control register
3. Read data from linear acceleration output registers and send them to PC via Serial communication
4. Upload and test your implementation moving with the sensor in different direction. (X, Y, Z axes are marked at IMU)
5. By experimenting guess the output value which corresponds into acceleration of 1g (9.81ms<sup>-1</sup>). Convert output value into unit of g.

## 6.8 MinIMU - Angular rate

Use MinIMU for measuring angular rate.

1. Open file **IMU\_ang\_rate.ino**
2. Follow the instructions inside and write your own code that sets up the IMU by writing setup value into control register
3. Read data from angular rate output registers and send them to PC via Serial communication
4. Upload and test your implementation by rotating IMU along different axes. (X, Y, Z axes are marked at IMU)



## 6.9 MinIMU - Angular rate integral

TODO

1. Open file **IMU\_ang\_rate\_integral.ino**
2. Follow the instructions inside and write your own code that sets up the IMU by writing setup value into control register
3. Read data from angular rate output registers and send them to PC via Serial communication
4. Upload and test your implementation by rotating IMU along different axes. (X, Y, Z axes are marked at IMU)

## 6.10 MinIMU - Temperature sensor

Use MinIMU to measure relative temperature.

1. Open file **IMU\_temperature.ino**
2. Follow the instructions inside and write your own code that reads data from temperature sensor and sends it to PC via Serial communication
3. Upload and test your implementation by touching the sensor (warming it up) and blowing air on it (cooling it down)
4. Bonus assignment: Calibrate the sensor.

## 6.11 MinIMU - Magnetometer

Use MinIMUs magnetometer for measuring magnetic fields around the sensor.

1. Open file **IMU\_magnetometer.ino**
2. Follow the instructions inside and write your own code that reads data from the magnetometer and sends it to PC via Serial communication
3. Upload and test your implementation by rotating the sensor. Sensor can even detect earth's magnetic field. Alternatively you can put to magnetometers close proximity item made out of iron, or speaker from a phone
4. Your assignment: Find NORTH

## 7 Appendix

mozno sem schemy zo vsetkych cviceni?

todo arduino mega scheme

ideas/feedback aby som nezabudol

V labu 1 bych přidal i info o VSCode a Platformio, Arduino IDE 2 a třeba i Arduino CLI, osciloskop

"v tomto díle uvidíte" a v dalších labinách i "v minulých dílech jste viděli"

spravit nejaký feedback papier na ktorý by na každom cviku napísali čo sa páčilo/bolo hard a pod