Measurements, Sensors and Data Logging Course

Week 5

Simple Datalogger

Lesson 8

Arduino Shields

Lesson 8: Simple Datalogger

- What is an Arduino Shield?
  - An Arduino shield is a circuit board that follows the basic shape and pinout of the Arduino board. It plugs into the top of the Arduino Board to extend the capabilities of the Arduino with the circuit on the shield.
- How do we use an Arduino Shield?
  - Arduino shields are plugged into the top of the Arduino board and can be stacked together.
  - Libraries and code can then be used to access the additional capabilities of the shield.
- More Information:
  - https://learn.sparkfun.com/tutorials/arduino-shields
  - https://www.arduino.cc/en/Main/arduinoShields

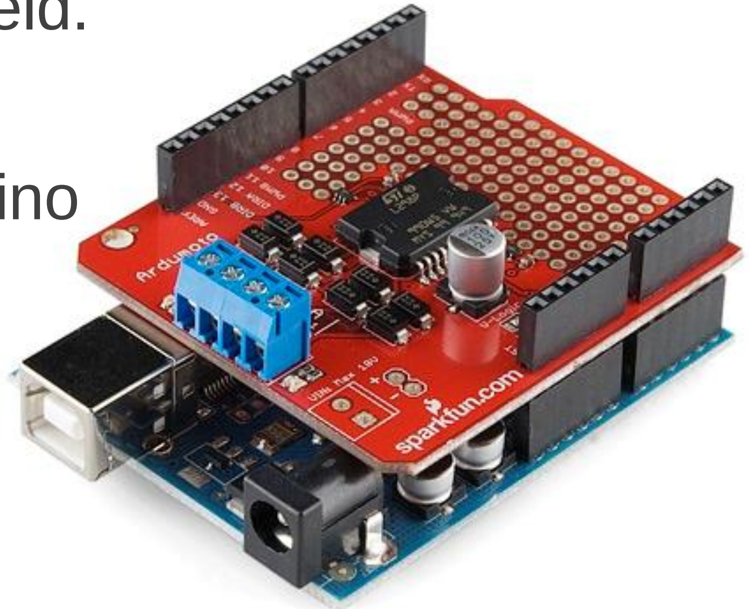


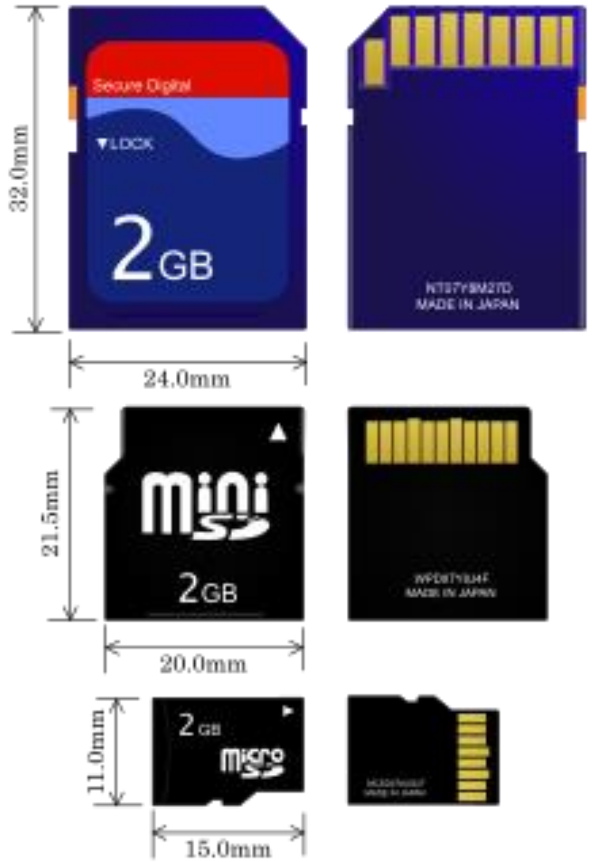
Image copied from https://learn.sparkfun.com/tutorials/arduino-shields

SD Card

Lesson 8: Simple Datalogger

- What is an SD Card?
  - Memory storage device
  - Stands for Secure Digital
  - Multiple Sizes: physical form factors and memory capacity
- Where is Flash data storage used?
- How do we use an SD card with an Arduino?
  - SD card shield!!
  - SD card library
  - SPI communication (digital communication)

  - https://www.arduino.cc/en/reference/SD

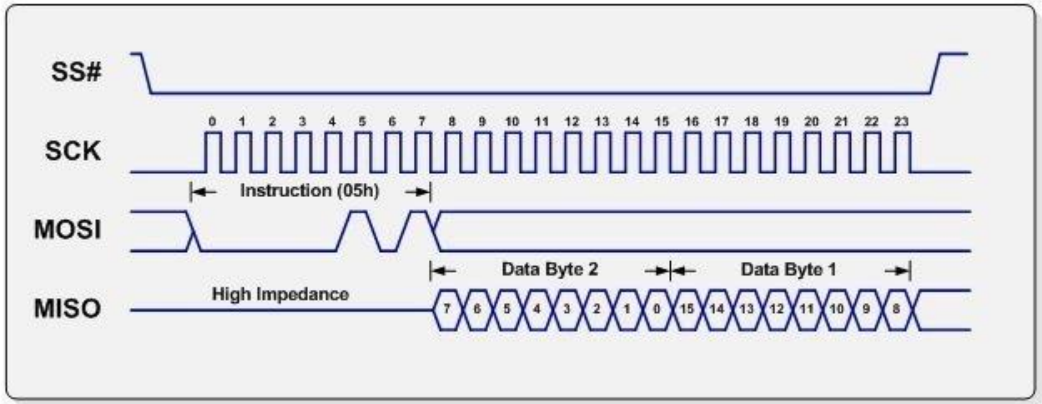


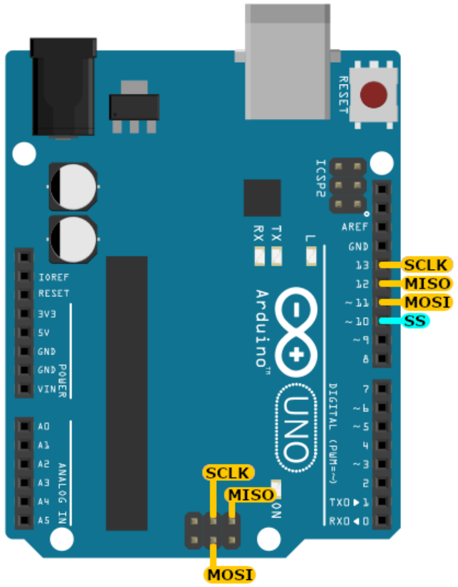
CSV File
Lesson 8: Simple Datalogger

- What is a CSV file?
  - Comma Separated Value
  - Typically used as a generic spreadsheet, can be opened in many programs.
    - Excel, Google Sheets
  - Also used for exporting and importing data with specialized programs.
- How do we use a CSV file?
  - Can be opened in Excel and used similar to a regular excel file
    - Excel features (math, etc) cannot be saved
    - File can be saved as a regular .xls / .xlsx
  - Can also be opened in google sheets, notepad++, etc

SPI
Lesson 8: Simple Datalogger

- SPI (Serial Peripheral Interface), multiple devices, short distances
  - Synchronous Communication: clock signal shared between master and slave
  - MISO: Master In Slave Out, *sends data to the master from the slave device*
  - MOSI: Master Out Slave In, *data from the master to the slave device*
  - SCK: Serial Clock
  - CS: Slave Select (individual for each device)
  - https://www.arduino.cc/en/reference/SPI
  - https://www.corelis.com/education/tutorials/spi-tutorial/





Master Device

Slave Device

Lesson 8 Hardware

Lesson 8: Simple Datalogger

- What hardware will we need for this Lesson?
  - Potentiometer on pin A0
  - Grove Light Sensor on pin A6
  - Grove Sound Sensor A2
  - Seeeduino Lotus (Arduino Uno compatible board)
  - SD Card Shield (HiLetGo)
    - Multiple options are available
  - SD Card
    - Formatted as FAT16

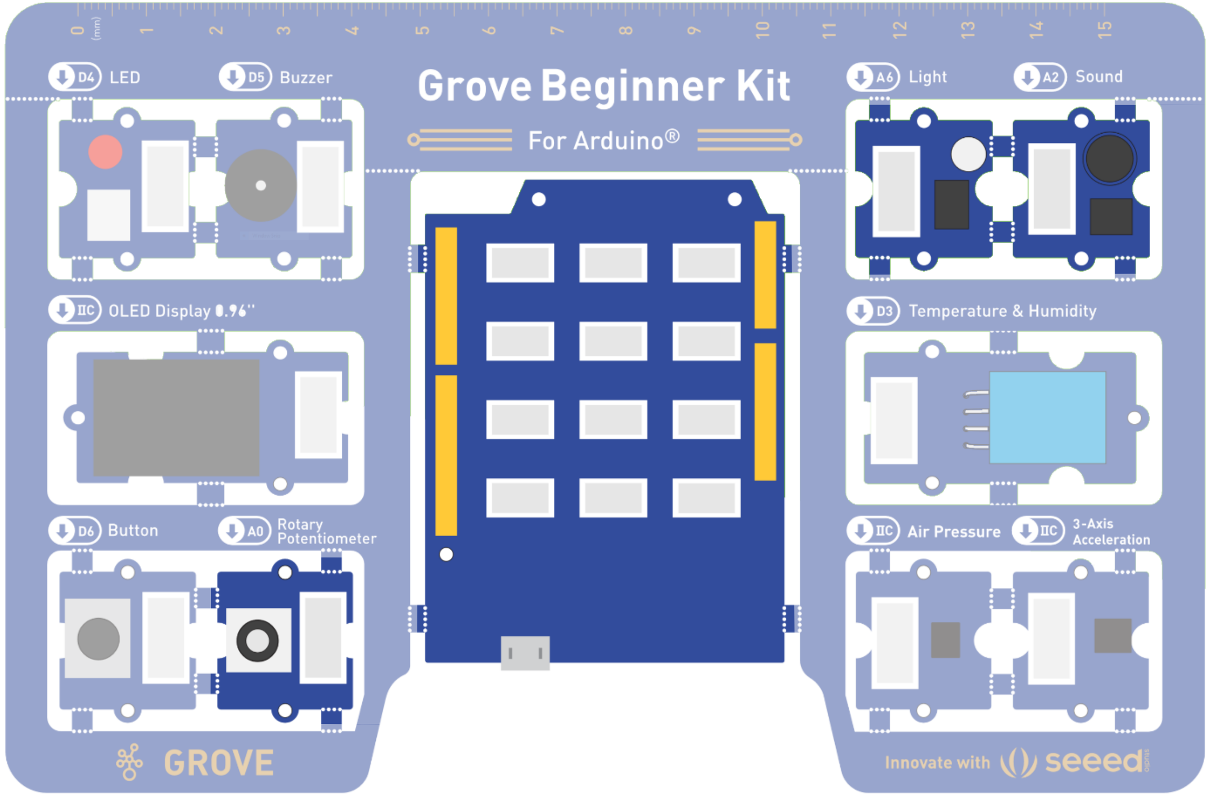


Image modified from https://files.seeedstudio.com/wiki/Grove-Beginner-Kit-For-Arduino/res/Grove-Beginner-Kit-For-ArduinoPDF.pdf

Image copied from https://www.amazon.com/HiLetgo-Logging-Recorder-Logger-Arduino/dp/B00PI6TQWO/

Image copied from https://www.microcenter.com/product/485234/micro-center-64gb-microsdxc-class-10--uhs-1-flash-memory-card

Assemble the Datalogging Hardware

Lesson 8: Simple Datalogger

1. Carefully align the pins of the shield with the sockets in the Seeeduino Lotus and press down until seated
2. Place micro-SD card into SD card adaptor and insert into SD Card Shield
3. Plug in USB cable between Seeeduino Lotus and Computer

Open and Upload Sketch

Lesson 8: Simple Datalogger

4. Open Simple Datalogger Sketch
   – **File → Sketchbook → CrashCourse_Jan→ L8_Simple_Datalogger.ino**
5. Upload the sketch to your Arduino by clicking the Upload Button.
   – The sketch should compile, and then upload to your Arduino.
6. Open the serial monitor.
   – **Tools → Serial Monitor** (Ctrl+Shift+M)
7. Observe the output in the Serial Monitor for a few seconds
   – Cover the light sensor, rotate the potentiometer, and make some noise to change the value of the sensor readings
8. Unplug the USB connector

Open Logged Data on Computer

Lesson 8: Simple Datalogger

9. Remove SD card from Shield
10. Insert into computer
11. Open “datalog.csv” in excel or other spreadsheet program
12. Bonus Activity: Graph the data!

SD Library

Lesson 8: Simple Datalogger

```
#include <SD.h>
```

- The SD Library allows us to communicate with SD cards formatted with FAT16 and FAT32 filesystems.
- It is built into the Arduino IDE, so there is no need to install it.
- It takes advantage of the SPI interface of the SD card, so the SPI library must also be included (`#include <SPI.h>`)before the SD library.
- It requires the chip select (sometimes called slave select) pin from the SD card to be connected to a digital output of the Arduino.
  - The SD Card Shield we chose uses pin D10 for the chip select
- More information:
  - https://www.arduino.cc/en/Reference/SD

SD Library – Appending data to a file

Lesson 8: Simple Datalogger

- We can append data to a file by:
  - Opening the file for writing
  - Checking that the file was opened
  - Writing a String to the file (we cover creating this String on the next slides)
  - Closing the file

```
File dataFile = SD.open("datalog.csv", FILE_WRITE);
if (dataFile)
{
  dataFile.println(dataString);
  dataFile.close();
}
```

Strings

Lesson 8: Simple Datalogger

```
String example = "This is an example of a string.";
```

- Create a String class instance named `example` containing the text "This is an example of a string."
- Strings are representations of **text** instead of numbers, and as such they do not behave in the same way as our previous datatypes.
- We have used string constants before in our sketches. Look for text encased in quotes, usually inside our `print()` and `println()` statements.
- More information:
  - https://www.arduino.cc/reference/en/language/variables/data-types/stringobject/
  - https://www.arduino.cc/en/Tutorial/BuiltInExamples#strings

Converting other datatypes to Strings
Lesson 8: Simple Datalogger

- Syntax:

```
String(val)
String(val, base)
String(val, decimalPlaces)
```

  - `val` – variable to format as a string
  - `base` - (optional) if `val` is an integer, what base should be used for the string representation. Options – `DEC` (default), `BIN`, or `HEX`
  - `decimalPlaces` – (optional) if `val` is a float, how many decimal places should be used. 2 is default
- For example:
  - `String(42)` → "42" // convert integer value to DEC
  - `String(42, BIN)` → "101010" // convert integer value to BIN
  - `String(42, HEX)` → "2A" // convert integer value to HEX
  - `String('x')` → "x" // convert character value to String
  - `String(0.12345)` → "0.12" // convert float value to DEC to 2 places
  - `String(0.12345, 5)` → "0.12345" // convert float value to DEC to 5 places
- This operation is called type casting

String Concatenation

Lesson 8: Simple Datalogger

- Concatenation – combining 2 strings into one larger string
- There are several way to concatenate strings in Arduino
- Syntax:

```
string1 += string2;
```

  – Appends `string2` to the end of `string1`

- Example:

```
String string1 = "1";
String string2 = "2";
string1 += string2; // string 1 now equals "12"
Serial.print(string1); // prints 12
```

- More information and more ways to concatenate strings:
  – https://www.arduino.cc/reference/en/language/variables/data-types/string/functions/concat/
  – https://www.arduino.cc/reference/en/language/variables/data-types/string/operators/concatenation/
  – https://www.arduino.cc/reference/en/language/variables/data-types/string/operators/append/

Creating a CSV Row with a String

Lesson 8: Simple Datalogger

```
String dataString = “”;
dataString += String(val1);
dataString += ",";
dataString += String(val2);
dataString += ",";
dataString += String(val3);
```

```
val1,val2,val3
```

	A	B	C
1	val1	val2	val3

Activities

Lesson 8: Simple Datalogger

- Read the temperature and humidity and light sensors overnight
  - Read the data and open it in a program (excel) to analyze and plot

Real Time Clock
+
Time Library

Lesson 10

Real Time Clock

Real Time Clock + Time Library

- What is a Real Time Clock (RTC)?
  - A clock that keeps track of the current time, independent from the Arduino.
    - On-board crystal oscillator.
    - Battery keeps the RTC powered even when the Arduino is powered down.
  - Arduino can be used to read this time and program.
- How do we use the RTC?
  - Time and PCF8523 Libraries
  - I2C communication
  - Need to set the time on the RTC the first time it is powered (already set on your hardware)
- More Information:
  - https://www.pjrc.com/teensy/td_libs_Time.html
  - https://www.pjrc.com/teensy/td_libs_DS1307RTC.html

Hz + Logging Rates

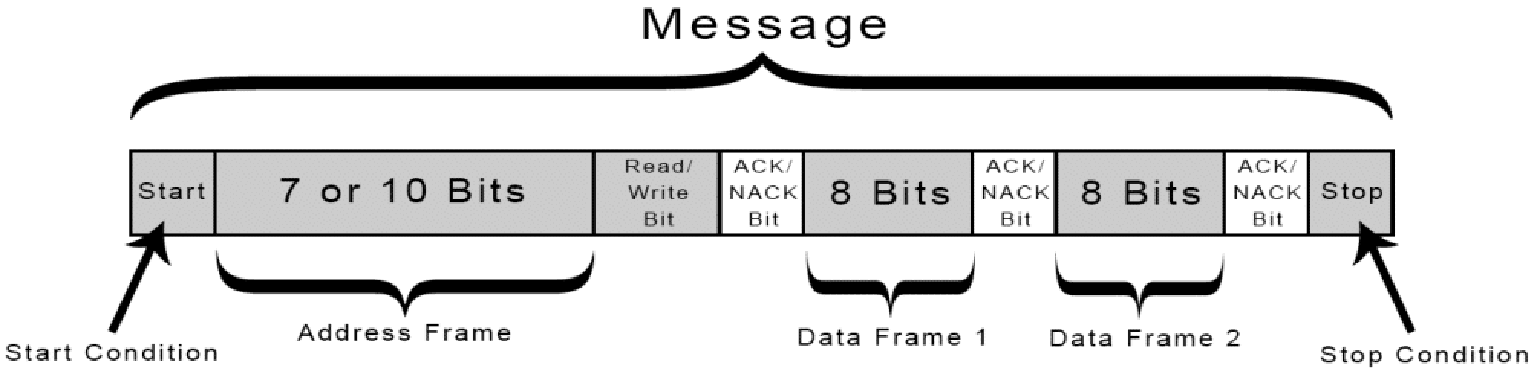
Real Time Clock + Time Library

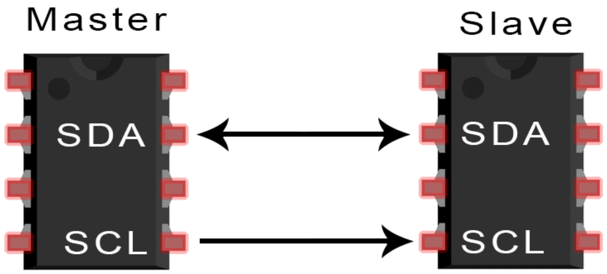
- What is Hz?
  - A unit of frequency. aka: how often something is happening each second
- How do we calculate Hz?
  - # of occurrences / second or 1 / time event takes (sec)
    - ${}^{\#\ of\ occurrences}/_{1\ (sec)}$ ex. 5 occurrences / sec = 5 Hz
    - ${}^{1}/_{time\ event\ takes\ (sec)}$ ex. 5 seconds / event => 1 / 5 = 0.2 Hz
- What rate should we log our data at?
  - Depends!
  - Ideal: 10 times faster than the signal changes
  - Minimum: 2 times faster than the signal changes

I²C (Inter-Integrated Circuit)

Real Time Clock + Time Library

- I²C, sometimes I2C or IIC
  - Two wires and robust, but slower transfer rate than SPI
  - Library: Wire.h
  - SDA (serial data): The line for the master and slave to send and receive data.
  - SCL (serial clock): The line that carries the clock signal.

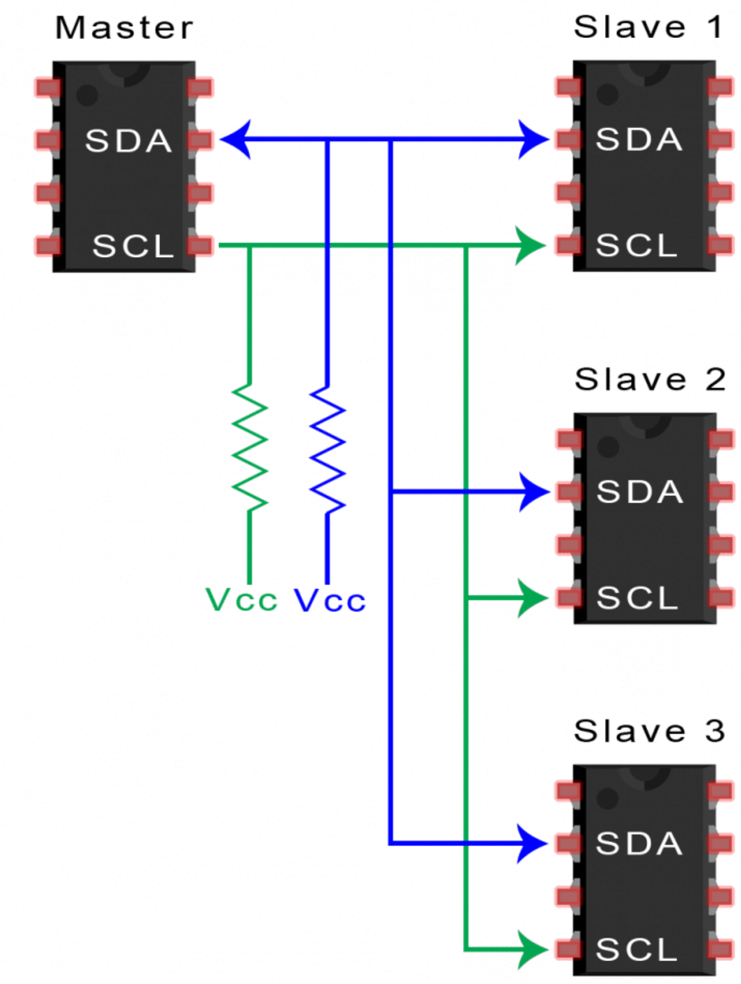


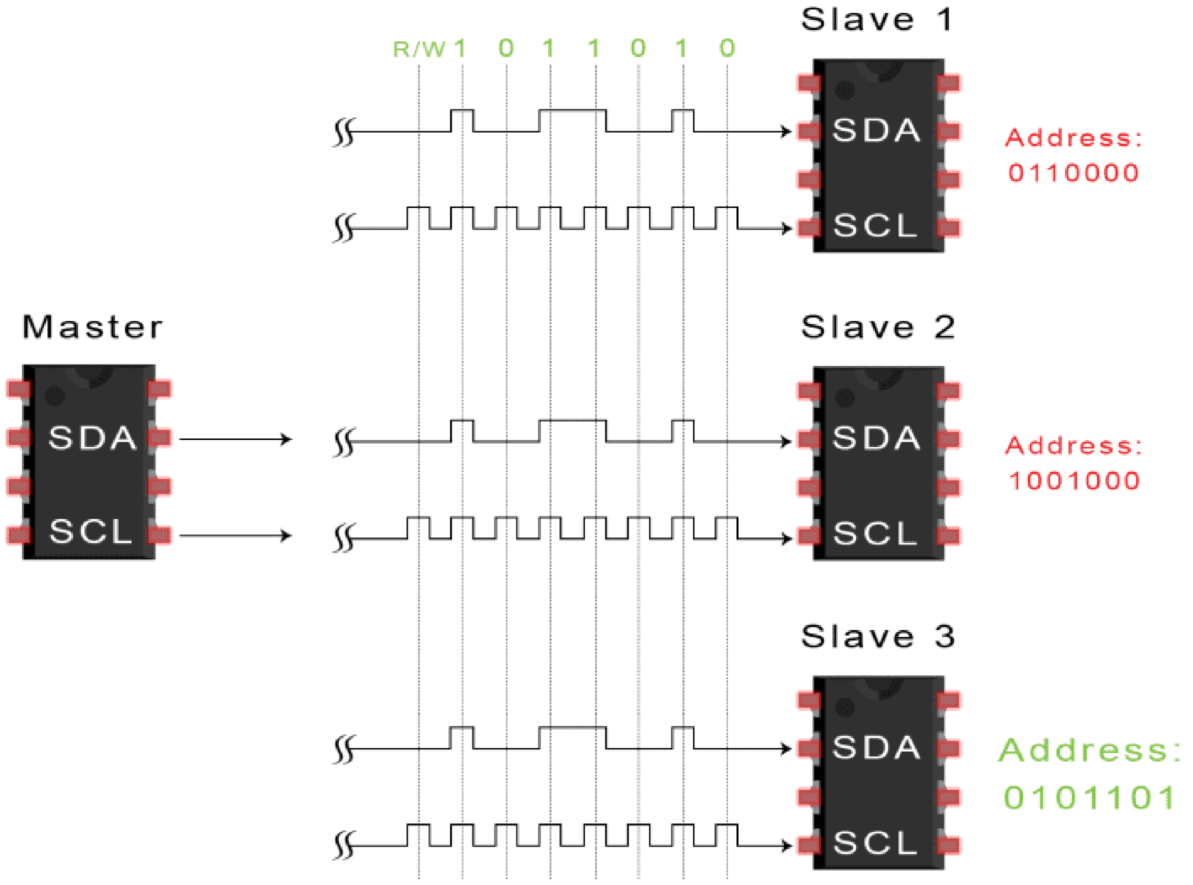


  - https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/#:~:text=I2C%20is%20a%20serial%20communication,always%20controlled%20by%20the%20master

I²C (Inter-Integrated Circuit)

Real Time Clock + Time Library





Lesson 9 Hardware
Real Time Clock + Time Library

- What hardware will we need for this Lesson?
  - Potentiometer on pin A0
  - Grove Light Sensor on pin A6
  - Grove Sound Sensor A2
  - Seeeduino Lotus (Arduino Uno compatible board)
  - SD Card Shield (HiLetGo) w/ battery
    - Multiple options are available
  - SD Card
    - Formatted as FAT16

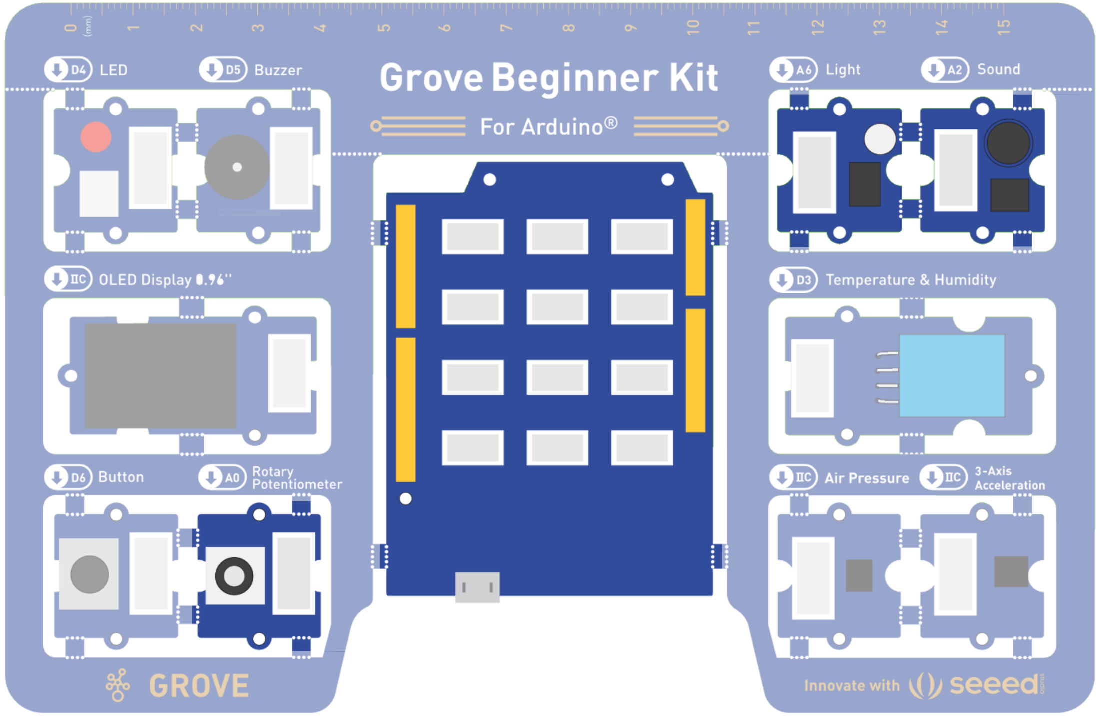


Image modified from https://files.seeedstudio.com/wiki/Grove-Beginner-Kit-For-Arduino/res/Grove-Beginner-Kit-For-ArduinoPDF.pd


Image copied from https://www.amazon.com/HiLetgo-Logging-Recorder-Logger-Arduino/dp/B00PI6TQWO/

Image copied from https://www.microcenter.com/product/485234/micro-center-64gb-microsdxc-class-10--uhs-1-flash-memory-card

Install the Time Libraries

Real Time Clock + Time Library

1. Install the Time library from the Library Manager
    a. Sketch → Include Library → Manage Libraries…
    b. Search for “**RTCLib**”
    c. Install **RTCLib** by Adafruit
    d. Search for “**TimeAlarms**”
    e. Install **TimeAlarms** by Michael Margolis
    f. Install **Time** by Michael Margolis

Open and Upload Sketch

Real Time Clock + Time Library

2. Open Simple Datalogger Sketch
   - **File → Sketchbook → FRSEF_Crash_Course → Week_4 → L9_Timed_Datalogger.ino**
3. Upload the sketch to your Arduino by clicking the Upload Button.
   - The sketch should compile, and then upload to your Arduino.
4. Open the serial monitor.
   - **Tools → Serial Monitor** (Ctrl+Shift+M)
5. Observe the output in the Serial Monitor for a few seconds
   - Cover the light sensor, rotate the potentiometer, and make some noise to change the value of the sensor readings
6. Unplug the USB connector

Open Logged Data on Computer

Lesson 8: Simple Datalogger

7. Remove SD card from Shield
8. Insert into computer
9. Open “datatime.csv” in excel or other spreadsheet program. Note the date and time recognized by the spreadsheet
10. Bonus Activity: Graph the data!

Time Library

Real Time Clock + Time Library

- What is the Time library?
  - An Arduino library that helps in keeping the time and provides functions to deal with seconds, minutes, hours, days, months and years.
  - The Time library can be synced with a RTC or other time and date services (GPS, NTP, Serial or other service that provides a standard Unix `time_t` time)

```
#include <TimeLib.h>
```

- Syntax:

  `year()` – return the current year

  `year(t)` – return the year of `t`

  `month()` – return the current month (1-12)

  `month(t)` – return the month of `t` (1-12)

  `day()` – return the current day of the month (1-31)

  `day(t)` – return the day of `t` (1-31)

  `hour()` – return the current hour (0-23)

  `hour(t)` – return the hour of `t` (0-23)

  `minute()` – return the current minute (0-59)

  `minute(t)` – return the minute of `t` (0-59)

  `second()` – return the current second (0-59)

  `second(t)` – return the second of `t` (0-59)

  `now()` – return the current time as a `time_t` number

  `t` is a `time_t` number

  `setSyncProvider(getTimeFunction)` – configure Time library to periodically call a user specified function to sync the clock. `getTimeFunction` is the name of the function that gets called.

- More Information
  - https://www.pjrc.com/teensy/td_libs_Time.html
  - https://github.com/PaulStoffregen/Time

RTCLib Library

Real Time Clock + Time Library

```
setSyncProvider(RTC.get);
```

- What is the RTCLib library?
  - An Arduino library to interface with the DS1307 RTC over I^2C (I2C or IIC).
  - It provides lower level access to the get (read) and set (write) the time to and from the RTC chip.

```
#include <RTCLib.h>
```

- Syntax:

  `rtc.now()` – reads the current date and time from the RTC and returns it as a `DateTime` number. (need to convert from DateTime to time_t to use inside the `setSyncProvider()` function from the time library.)

- More Information
  - https://github.com/adafruit/RTClib

RTCLib Library Setting the Time

Real Time Clock + Time Library

- How to set the time of the RTC (we have already done this for you)
  1. Open the SetTime example:
     a. File → Examples → RTCLib → pcf8523
  2. Upload to the Arduino
- This example utilizes the compile time to determine the current time and set it into the RTC.
- When would you have to do this:
  – After turning on the RTC after any power loss (ex battery died, first power on)
  – If the time is significantly off (remove and replace the battery in the shield while Arduino is unpowered to reset the clock)
  – Adjusting to a different time zone

TimeAlarms Library

Real Time Clock + Time Library

- What is the TimeAlarms library?
  - An Arduino library designed to work with the Time library to run functions at specific times.
  - Can work similar to an alarm clock or a timer and can trigger these alarms once or repeatedly.

```
#include <TimeAlarms.h>
```

- Syntax:

`Alarm.delay(milliseconds)` – check if alarm or timer should run and then delay for specified `milliseconds`

`Alarm.alarmRepeat(dayofweek, hours, minutes, seconds, function)` – create repeating alarm that calls `function` at a particular time. Optional `dayofweek` can be used to only repeat on a certain day.

`Alarm.alarmOnce(dayofweek, hours, minutes, seconds, function)` – create one off alarm to call `function` at a particular time. Optional `dayofweek` can be used to only trigger on a certain day.

`Alarm.timerRepeat(seconds, function)` – create a timer that calls `function` at `seconds` interval.

`Alarm.timerOnce(seconds, function)` – create one off timer to call `function` at a particular time once.

- More Information
  - https://www.pjrc.com/teensy/td_libs_TimeAlarms.html
  - https://github.com/PaulStoffregen/TimeAlarms

Activities

Real Time Clock + Time Library

- Read the temperature and humidity and light sensors at 1 minute intervals overnight
  - Read the data and open it in a program (excel) to analyze and plot
- Turn on the LED for 2 minutes at 8:00 AM and 8:00 PM each day

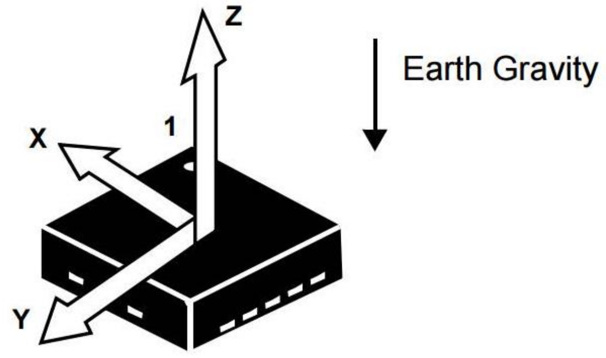
Accelerometer

Lesson 10

Motion Sensors

Accelerometer

- Accelerometers are used to measure acceleration, in linear directions
- Measurements in m/s^2
  - 1g = 9.8 m/s^2
  - 0g = object not moving or in free fall
- Types: analog, digital (IIC, SPI), PWM
- Uses
  - Position tracking
  - Force measurement
  - Vibration measurement



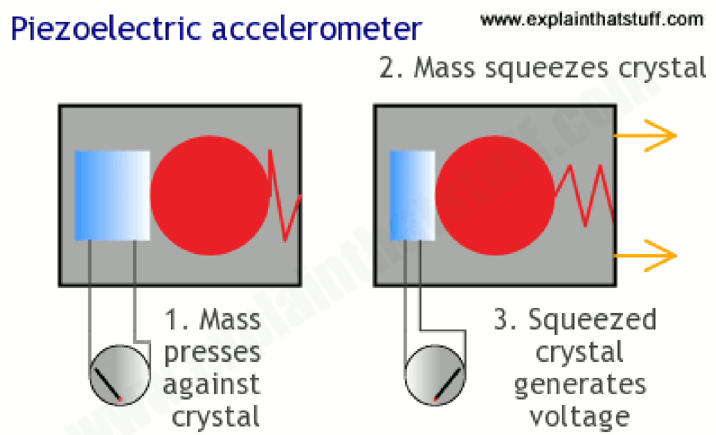
(TOP VIEW)
DIRECTION OF THE
DETECTABLE ACCELERATIONS

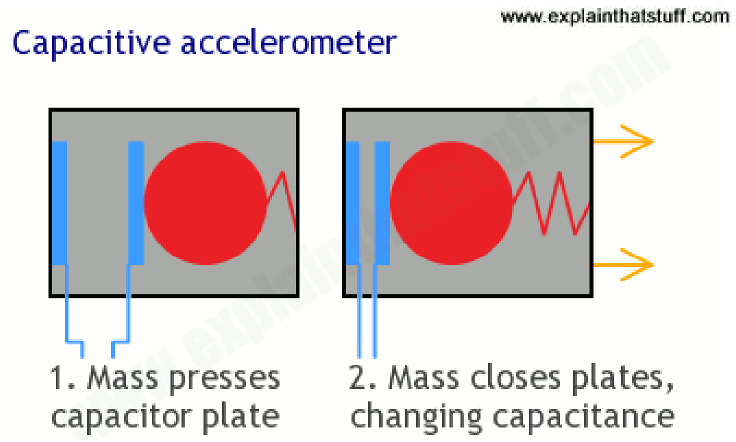
- https://www.adafruit.com/category/521
- https://learn.sparkfun.com/tutorials/accelerometer-basics/all
- https://www.sparkfun.com/pages/accel_gyro_guide?_ga=2.260802829.124947883.1606791953-761679650.1605223049
- https://www.seeedstudio.com/blog/2019/12/24/what-is-accelerometer-gyroscope-and-how-to-pick-one/

Motion Sensors

Accelerometer

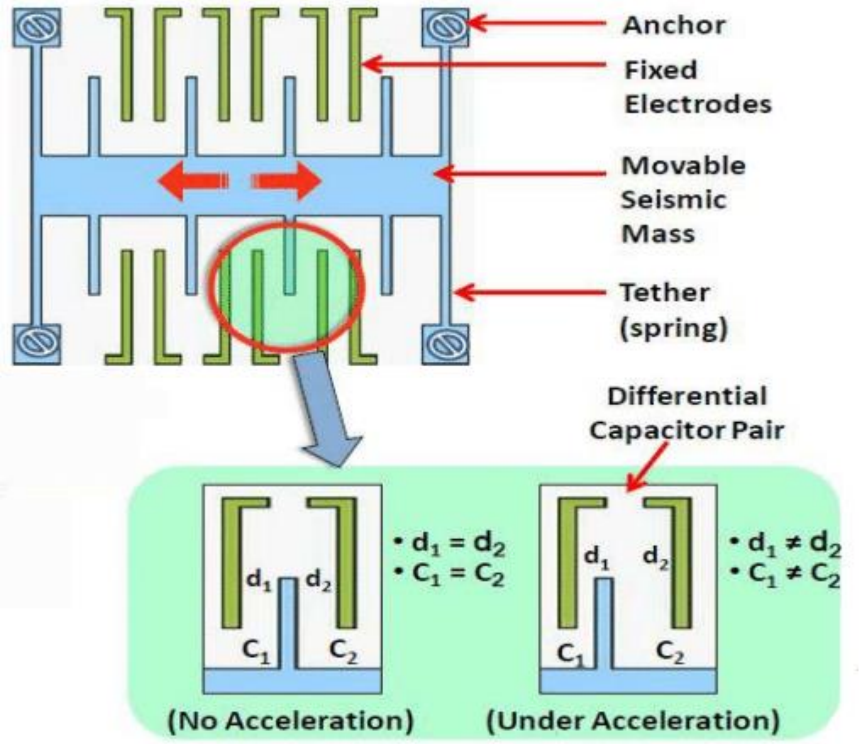
- Accelerometers are used to measure acceleration, in linear directions
- How they work
  - Capacitive: capacitive plates internally, some fixed and others on springs, motion between plates causes change in capacitance
  - Piezo Electric: Small mass on springs around piezo-electric materials. Electrical charges are created.





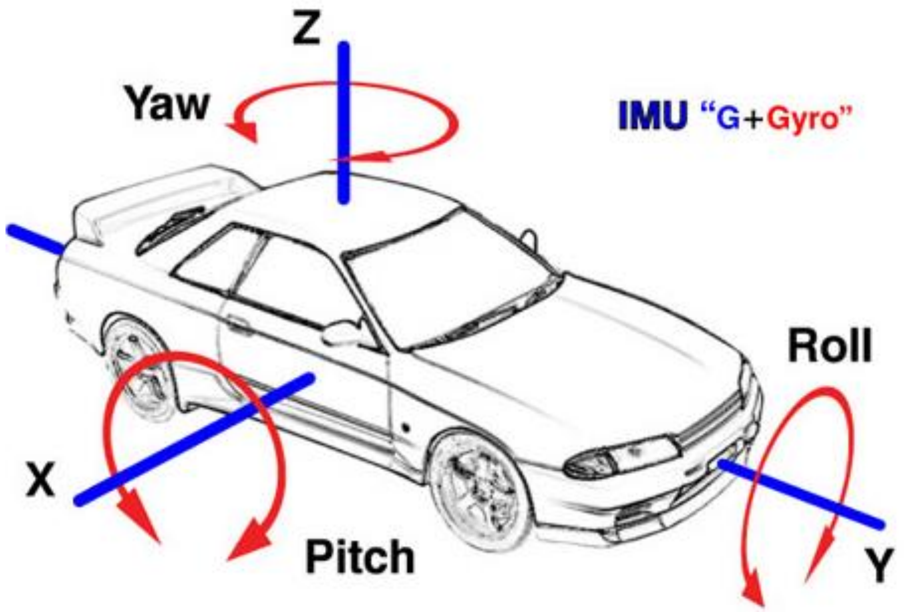
Motion Sensors
Accelerometer

- How they work
  - MEMS: microscopic, silicon based moving mass
    - Uses either piezo or capacitive changes
    - Cool Graphic



Motion Sensors
Accelerometer

- Other commonly used motion sensors:
  – Gyroscope: measure rotational motion
  – Magnetometer: measure magnetic force, typically magnetic north
- IMU: Accelerometer + Gyroscope



Hardware
Accelerometer

- What hardware will we need for this Lesson?
  - Grove 3-axis Accelerometer Module on IIC
  - Seeeduino Lotus (Arduino Uno compatible board)
  - SD Card Shield + SD Card
- Please assemble parts the same way we did last week

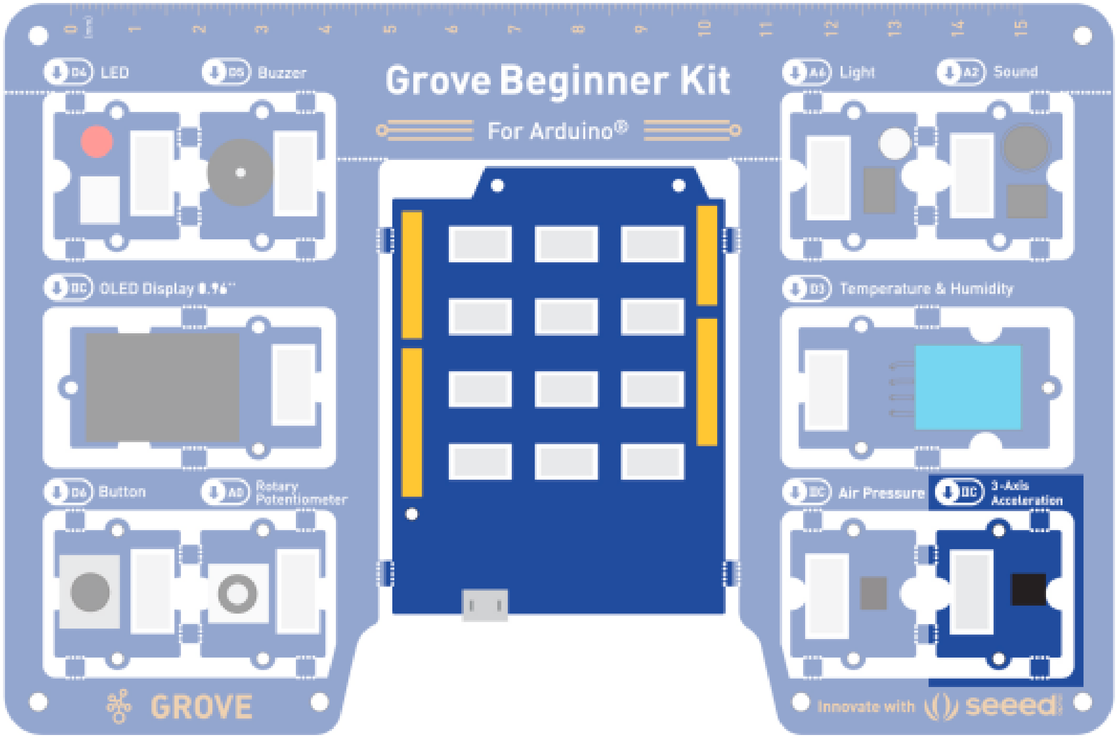


Image copied from https://www.amazon.com/HiLetgo-Logging-Recorder-Logger-Arduino/dp/B00PI6TQWO/

Image copied from https://www.microcenter.com/product/485234/micro-center-64gb-microsdxc-class-10--uhs-1-flash-memory-card

Library

Accelerometer

- Library to use: *sparkfun LIS3DH*
  - Search for ***Sparkfun LIS3DH*** in the library manager and install it.
  - There are multiple variants available for the LIS3DH sensor.
  - `#include <SparkFunLIS3DH.h>`
  - `LIS3DH myIMU(I2C_MODE, 0x19);`
- More Information:
  - https://github.com/sparkfun/SparkFun_LIS3DH_Arduino_Library

Open and Upload Sketch

Lesson 10: Accelerometer

1. Open Simple Datalogger Sketch
   - **File → Sketchbook → CrashCourse_Jan → L10_Accelerometer.ino**
2. Upload the sketch to your Arduino by clicking the Upload Button.
   - The sketch should compile, and then upload to your Arduino, assuming you have the correct COM port
3. Move the board around so that different faces point down for a couple seconds then gently shake or drop your board (from a couple inches)
4. Unplug the USB connector
5. Remove SD card and plug into your computer
6. Look at the data in your spreadsheet program
   - note the name of the file is now a number repressing the day, hour, minutes and seconds that the log started at. This prevents us from overwriting or appending data to older files!

Activity
Accelerometer

- With SD Card shield attached, start recording and conduct multiple drop tests.
  - Conduct test at different logging rates.
  - Be careful as the SD card may lose connection
- When complete, unplug the USB cable and remove the SD card
- Insert the SD card into your PC and plot the data

Example Datalog

Lesson 10: Accelerometer

	A	B	C	D	E
1	Accelerometer Log				
2	########				
3	Time (ms)	X (g)	Y (g)	Z (g)	Mag (g)
4	144	-0.0625	-0.025	1.0375	1.039681
5	184	-0.075	-0.0125	1.0375	1.040282
6	224	-0.0625	-0.0125	1.025	1.02698
7	264	-0.075	0	1.025	1.02774
8	304	-0.075	0	1.0125	1.015274
9	344	-0.075	-0.0125	1.0125	1.015351
10	384	-0.075	0	1.0375	1.040207
11	424	-0.1	0	1.0375	1.042308
12	464	-0.125	-0.05	1.05	1.058596
13	504	-0.125	-0.025	1.05	1.05771
14	544	-0.125	-0.0125	1.025	1.032669
15	584	-0.1	-0.0125	1.0125	1.017503
16	624	-0.1	0	1.0125	1.017426
17	664	-0.0625	-0.025	1.0125	1.014735
18	704	-0.075	-0.0375	1.025	1.028424
19	744	-0.075	-0.0125	1.0125	1.015351
20	784	-0.075	-0.0125	1.025	1.027816
21	824	-0.075	-0.0125	1.025	1.027816
22	864	-0.0875	0	1.025	1.028728
23	904	-0.075	0	1.0125	1.015274
24	944	-0.1	0	1.025	1.029866
25	984	-0.1125	0	1.0375	1.043582
26	1024	-0.075	-0.0125	1.05	1.052749
27	1064	-0.0375	-0.0375	1.0375	1.038854
28	1104	-0.0625	-0.025	1.025	1.027208
29	1144	-0.0625	0	1.025	1.026904
30	1184	-0.075	-0.0125	1.025	1.027816
31	1224	-0.0875	0	1.025	1.028728
32	1264	-0.1	0	1.025	1.029866
33	1304	-0.075	-0.0125	1	1.002887
34	1344	-0.0625	-0.0125	1	1.002029
35	1384	-0.0875	-0.0375	1	1.004521
36	1424	-0.1125	-0.025	1.0375	1.043881
37	1464	-0.1	-0.025	1.025	1.03017

09233437

Software Settings
Accelerometer

```
myIMU.settings.adcEnabled = 0;
myIMU.settings.tempEnabled = 0; // set to 1 to enable temp sensor readings
```

- Turn the output of the accelerometer on or off (1 = on)

```
myIMU.settings.accelSampleRate = sampleRate; // Hz.Can be:0,1,10,25,50,100,200,400,1600, 5000 Hz
```

- Sensor sample rates: how fast will the sensor update its output (`0,1,10,25,50,100,200,400,1600,5000 Hz`)

```
myIMU.settings.accelRange = 16;   // 16 to read the sudden stop at the end of a drop, max G readable.
```

- Range of the accelerometer (options: 2, 4, 8, 16)
  - 2: -2g to +2g (-19.6 m/s^2 to +19.6 m/s^2)
  - 16: -16g to +16g (-156.8 m/s^2 to +156.8 m/s^2)

```
myIMU.settings.xAccelEnabled = 1;
myIMU.settings.yAccelEnabled = 1;
myIMU.settings.zAccelEnabled = 1;
```

- Turn the output of the accelerometer on or off (1 = on)

https://github.com/sparkfun/SparkFun_LIS3DH_Arduino_Library

Reading the Accelerometer

Accelerometer

`myIMU.readFloatAccelX()`

– returns a floating point number of the acceleration in g's for the X axis

`myIMU.readFloatAccelY()`

– returns a floating point number of the acceleration in g's for the Y axis

`myIMU.readFloatAccelZ()`

– returns a floating point number of the acceleration in g's for the Z axis

- More Information:
  - https://github.com/sparkfun/SparkFun_LIS3DH_Arduino_Library

Code Analysis: Dynamically creating a filename

Accelerometer

- `createFilename()`
  - Function that uses the Time from the RTC to create a unique filename each time it is called
  - Stores the filename in a global String variable called `filename`
  - Filename is formatted as DDHHMMSS.csv using the time that the file was created.
    - You can change this behavior if you need a different format say YYYYMMDD if you are creating a file every few days or so.
  - Log the data several times to see this function in action! You should find several files on your SD card.

Sensors & Applications

Sensors & Applications – EEG, ECG EMG

- Measures of biopotential, the electrical output of human activity
  - Electroencephalogram (EEG)
    - Monitors brain activity
    - Measurements at forehead, top of head (potentially) and ears
  - Electrocardiogram (EKG)
    - Measures heart activity
    - Measurements at torso, arms and legs
  - Electromyography (EMG)
    - Electrical activity of muscles
    - Common test is to measure muscle response relative to stimulation of the muscle, measure a specific muscle

https://www.sensortips.com/featured/what-is-the-difference-between-an-ecg-eeg-emg-and-eog/

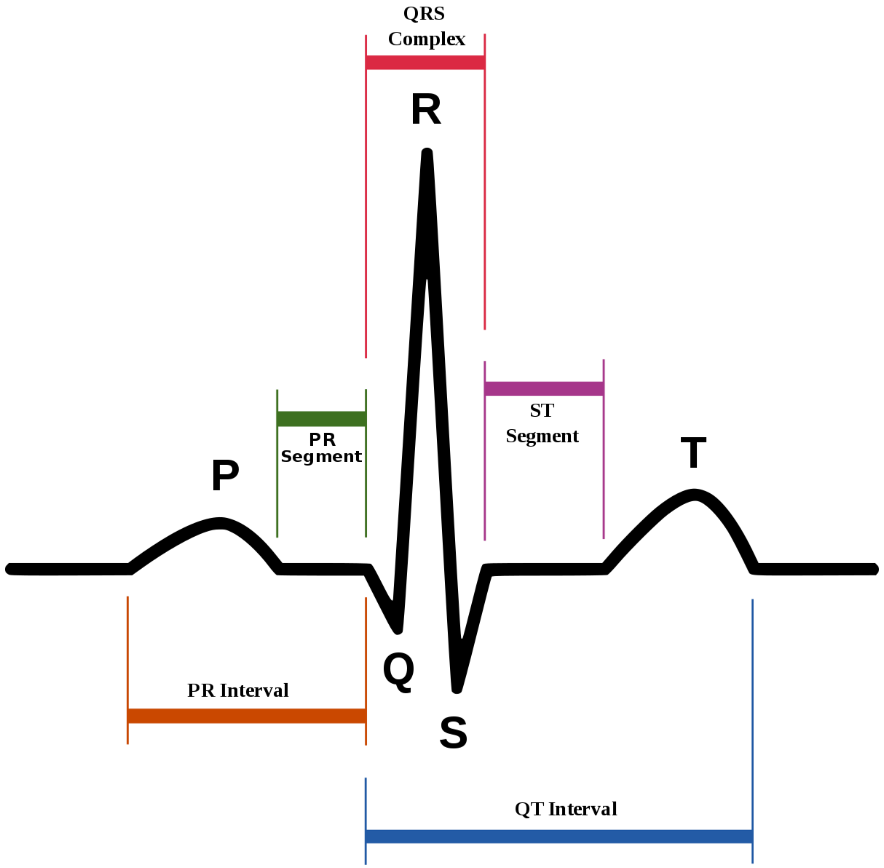
https://www.withings.com/de/en/health-insights/about-ecg-ekg-electrocardiogram

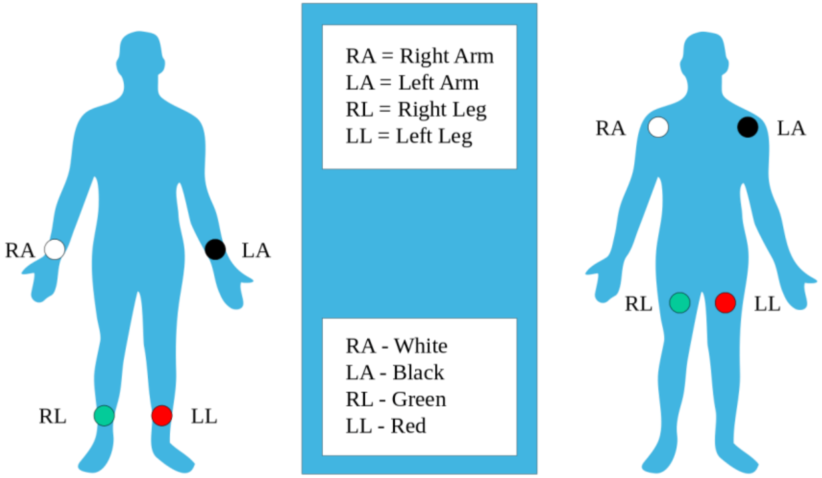
Sensors & Applications – EEG, ECG EMG

- Measuring
  - Amplifier is required (very lower voltages)
  - Electrodes used to “pick up” the voltages

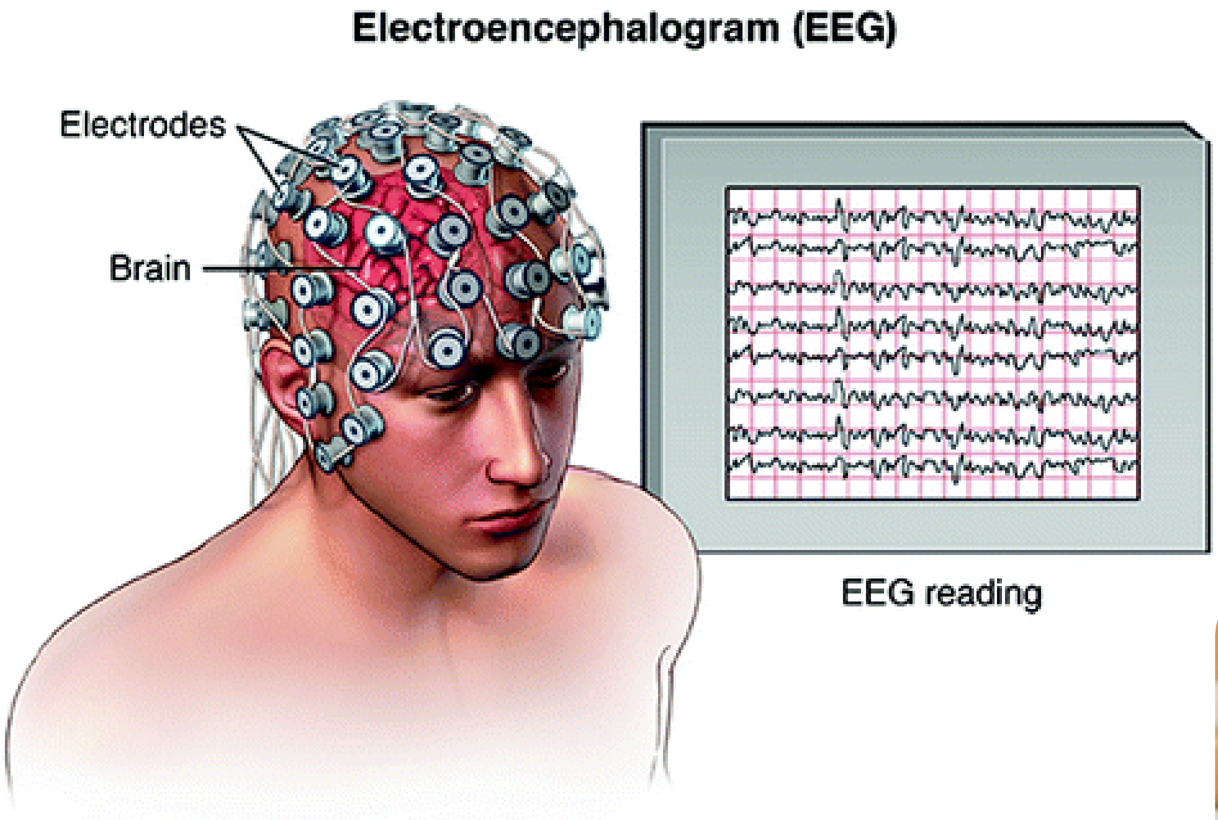
Source	Amplitude (mV)	Bandwidth (Hz)
ECG	1-5	0.05-100
EEG	0.001-0.01	0.5-40
EMG	1-10	20-2000
EOG	0.01-0.1	dc-10

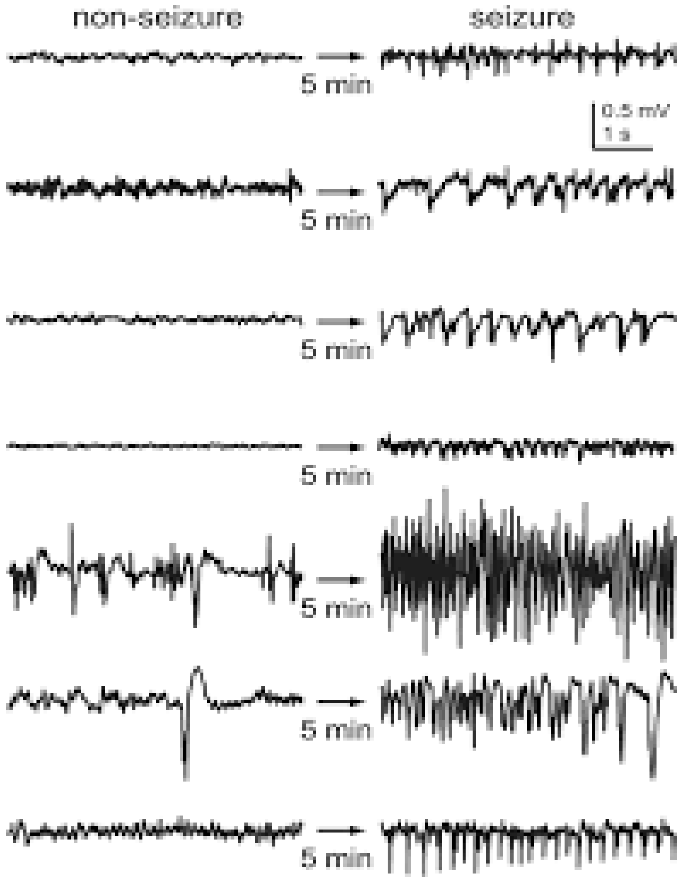
Sensors & Applications – EKG



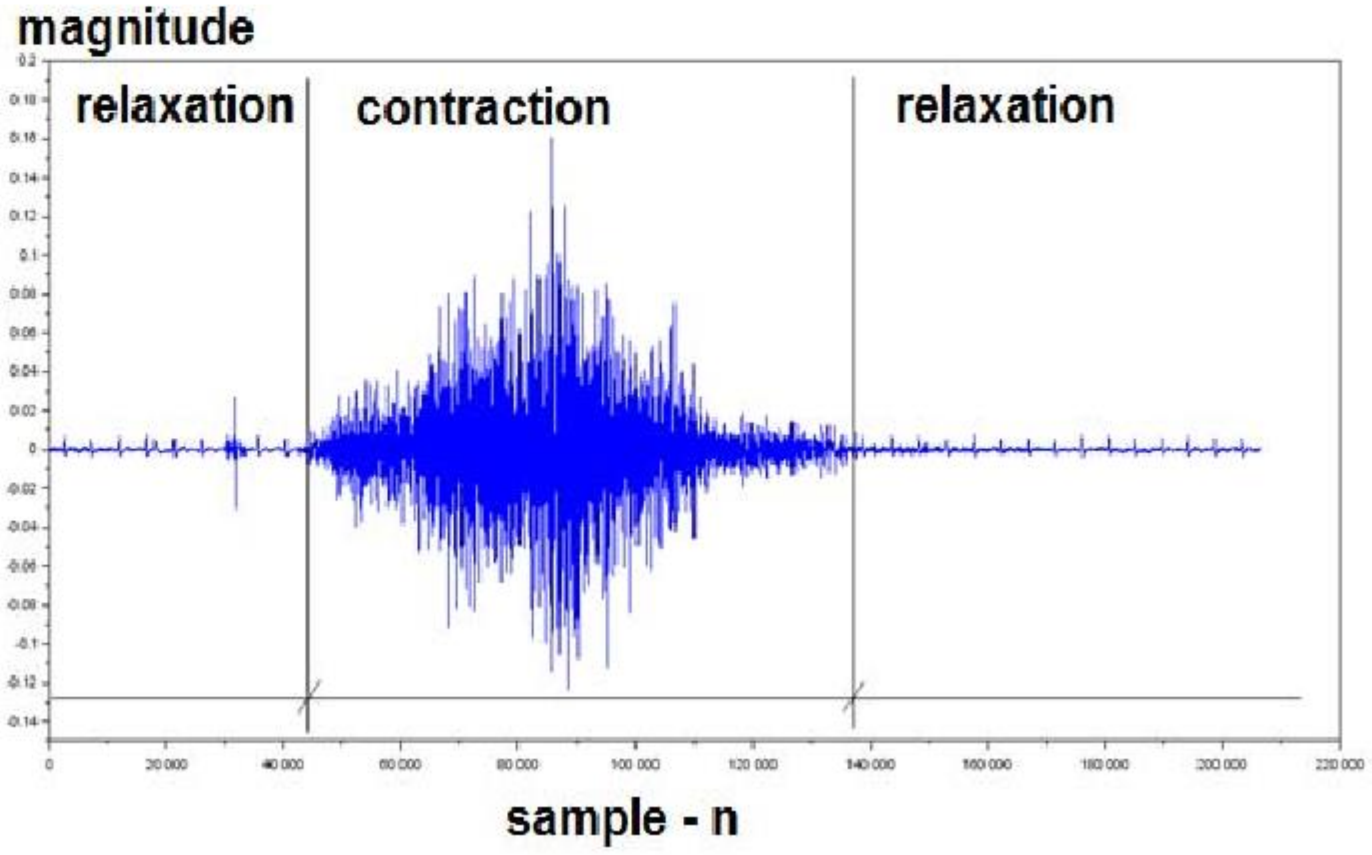


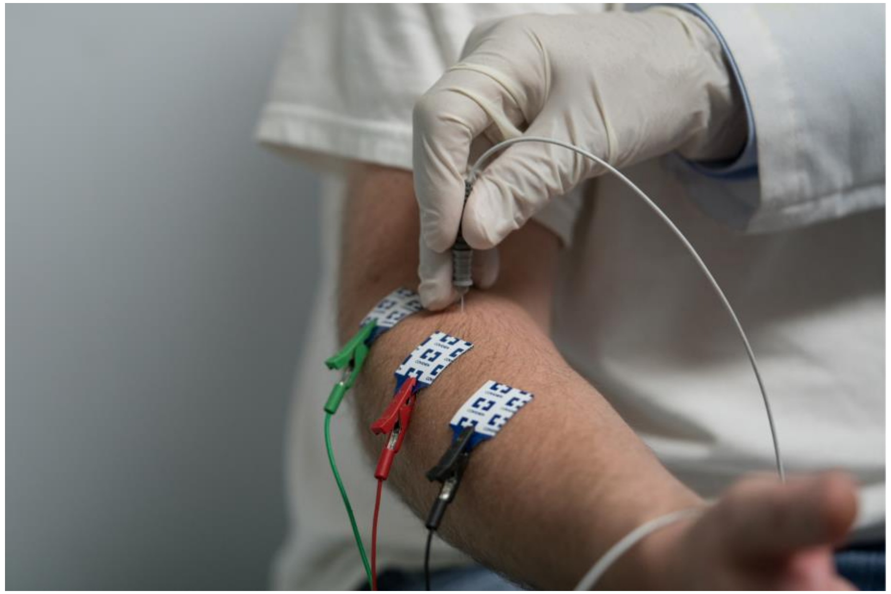
Sensors & Applications – EEG



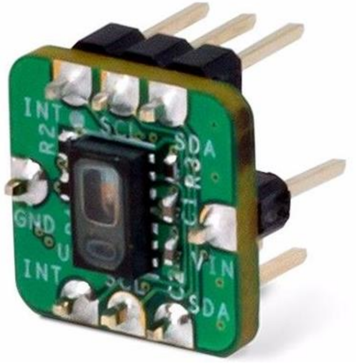


Sensors & Applications – EMG

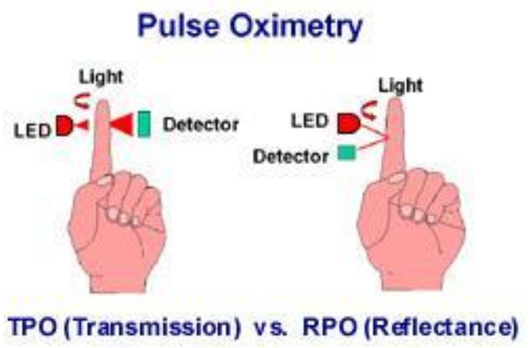




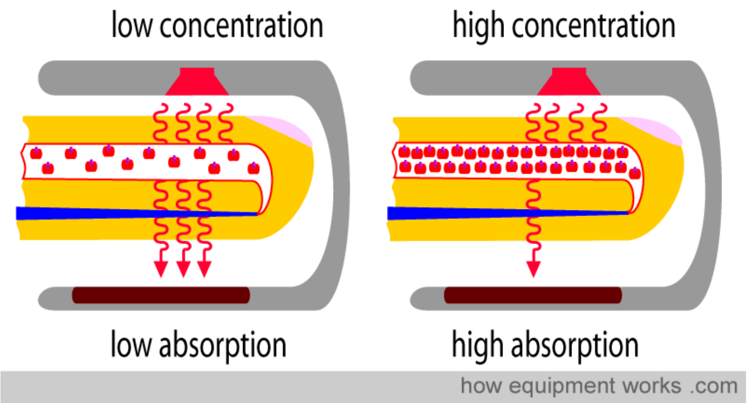
Sensors & Applications – Pulse Ox

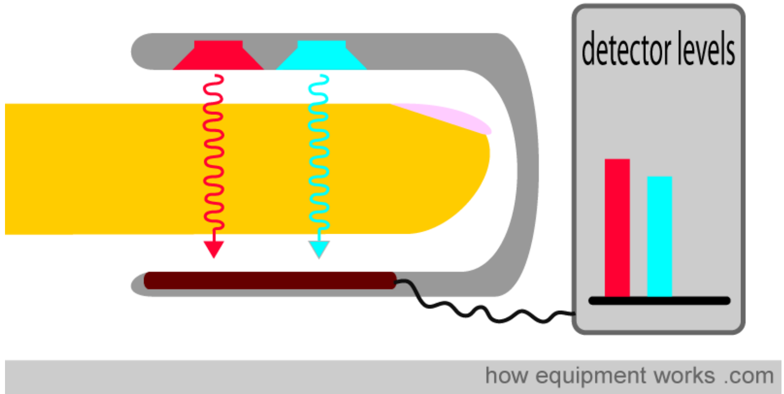


- Pulse-Oximetry
  - Measure blood oxygen saturation (Sp02) and calculate heart rate
    - Oxygen molecules attach to hemoglobin
    - Types: Transmission and Reflectance
    - Hemoglobin with and without oxygen absorbs light differently (wavelength of light differs)
      - Oxy Hb absorbs more infrared light than red light
      - Deoxy Hb absorbs more red light than infrared light

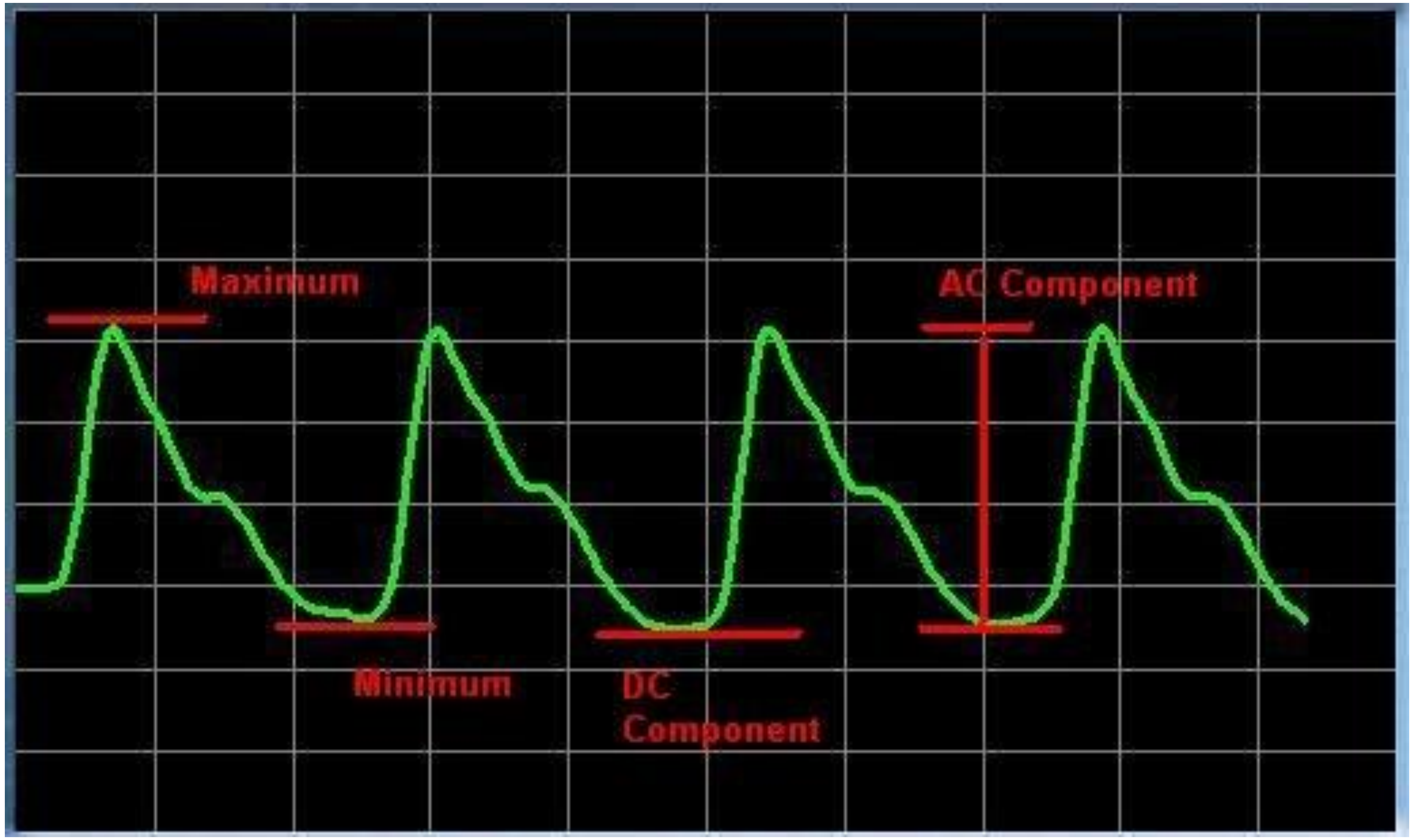


https://www.howequipmentworks.com/pulse_oximeter/





Sensors & Applications – Pulse Ox

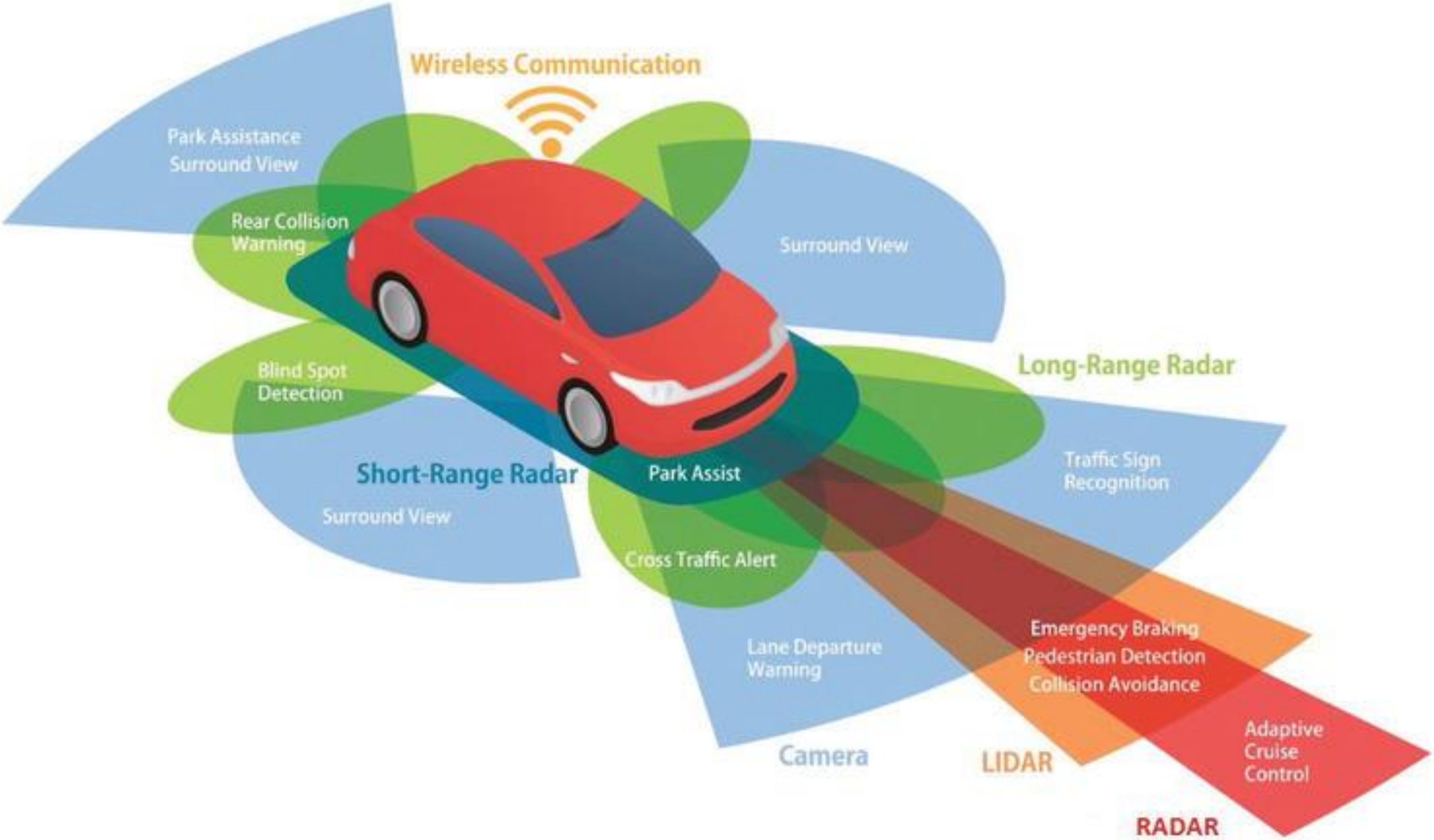


Sensors & Applications

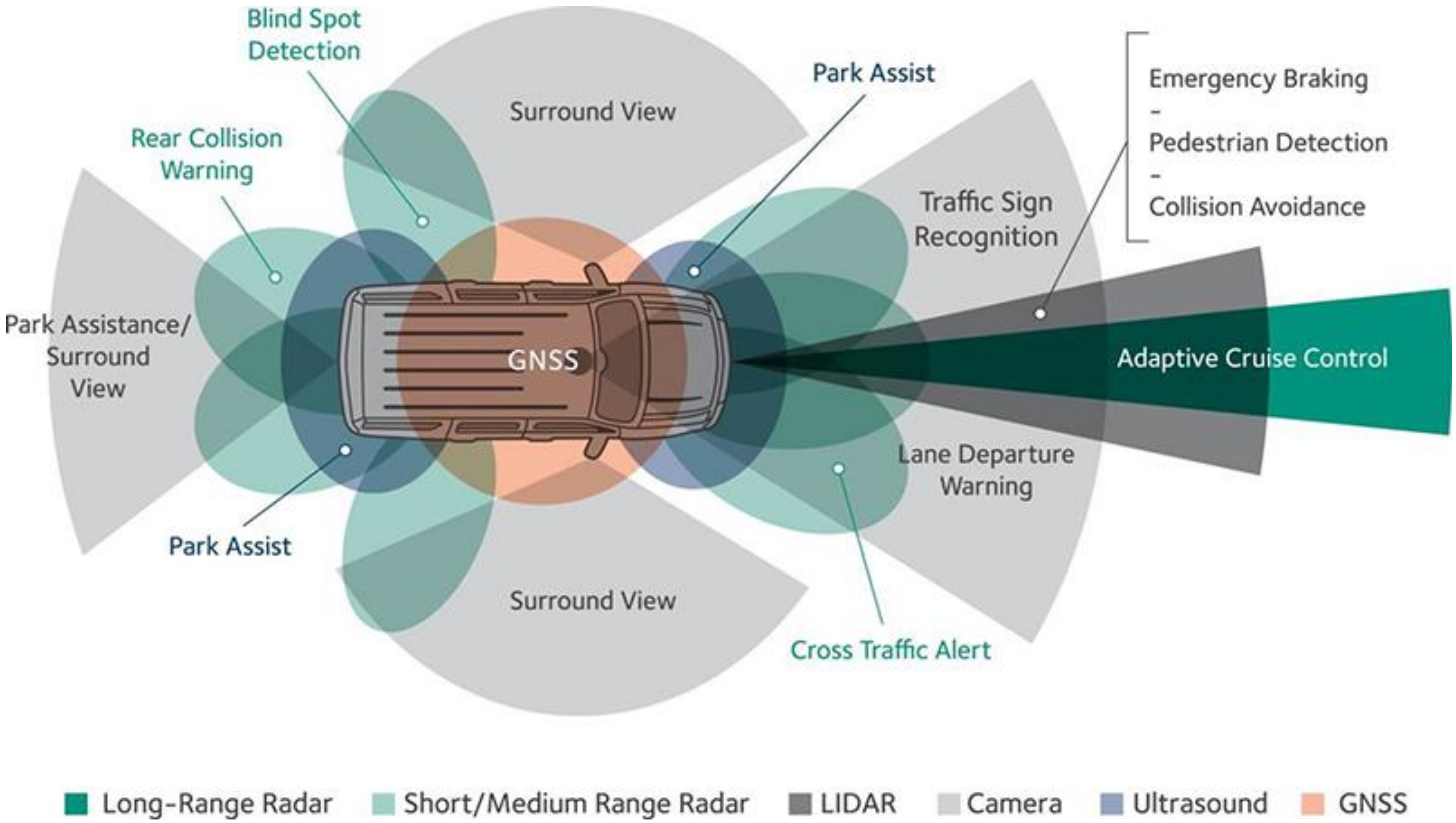
Sensors & Applications: Autonomous Driving

- Autonomous Driving
  - IMU (accel, gyro)
  - Location
    - GNSS (Global Navigation Satellite System) or GPS
  - External Object Detection
    - Radar
    - Lidar
    - Ultrasonic
    - Cameras
  - Vehicle Sensors
    - Wheel speed, steering angle

Sensors & Applications: Autonomous Driving



Sensors & Applications: Autonomous Driving



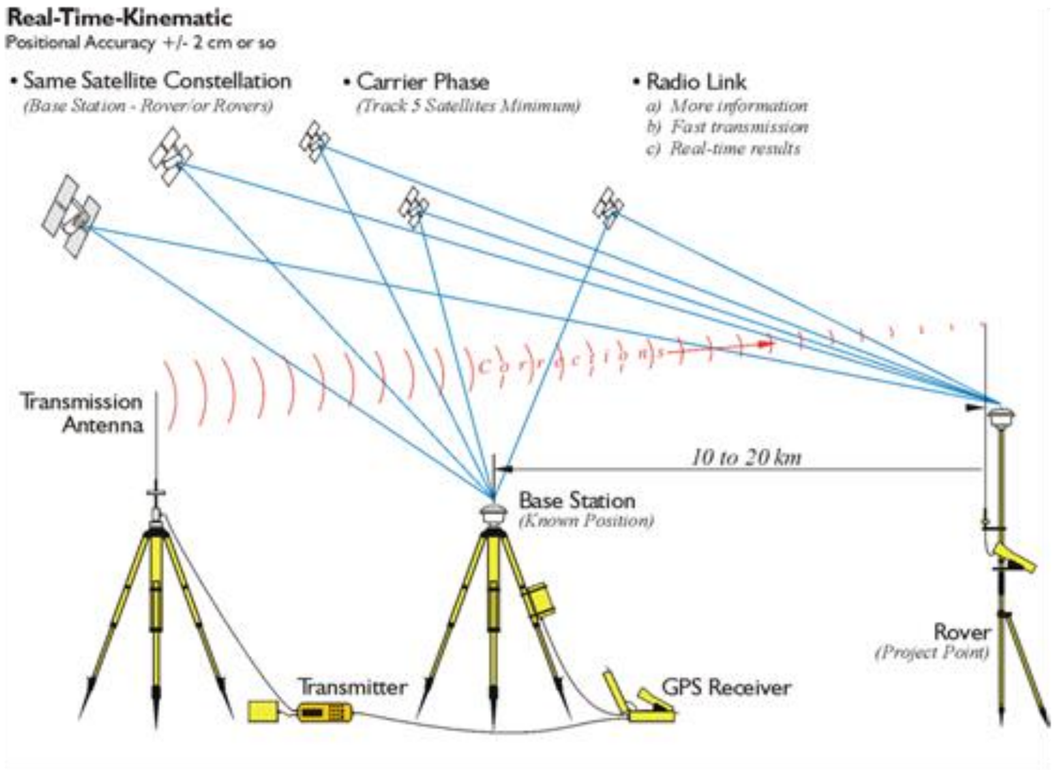
Sensors & Applications: Autonomous Driving

- GNSS (Global Navigation Satellite System)
  - Multiple systems in opreration
    - GPS (US), Galileo (EU), GLONASS (Russia), Beidou (China)
  - Time clock
  - Update rate limitations
    - Common: 10 – 20 Hz
    - Mobile Phone: 1 Hz
    - Use IMU data to calculate location between GPS updates (gps imu fusion)
    - Location Calculation: integrate acceleration to get location, small errors start to add up (drift)

GPS Update	IMU Calc	IMU Calc	IMU Calc	IMU Calc	IMU Calc	IMU Calc	IMU Calc	IMU Calc	IMU Calc	GPS Update	IMU Calc
0ms	10ms	20ms	30ms	40ms	50ms	60ms	70ms	80ms	90ms	100ms	110ms

Sensors & Applications: Autonomous Driving

- GNSS (Global Navigation Satellite System)
  - Correction services: clock drift, orbit errors, atmosphereic errors
    - PPP: Precise Point Positioning – global correction
    - RTK: Real Time Kinematic – local base station that offset data
    - DGPS: Differential GPS – base station that provides offset data



https://www.taoglas.com/centimeter-level-positioning-with-single-band-rtk/

https://www.e-education.psu.edu/geog862/node/1828

Sensors & Applications: Autonomous Driving

- Radar
  - Output radio waves and measure signals received back.
  - Short Range Radar: ~24 GHz
    - 1 to 30 m
  - Long Range Radar: ~77 GHz
    - 3 to 80 – 200m
  - Pros: Robust for weather, low cost
  - Cons: Lower resolution

Sensors & Applications: Autonomous Driving

- Radar examples

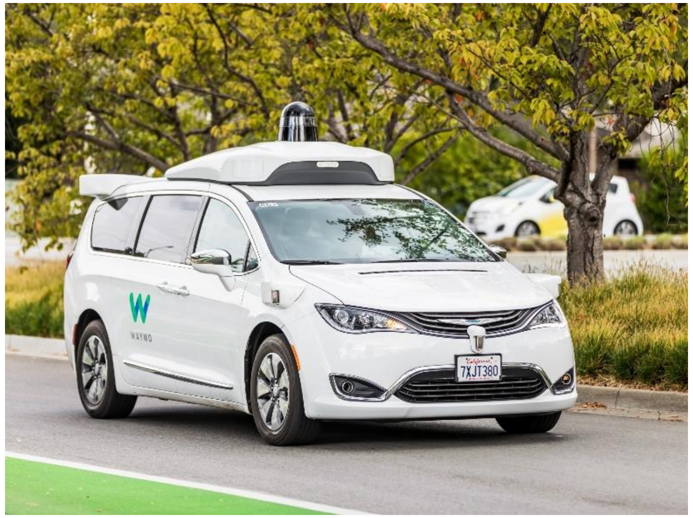
Sensors & Applications: Autonomous Driving

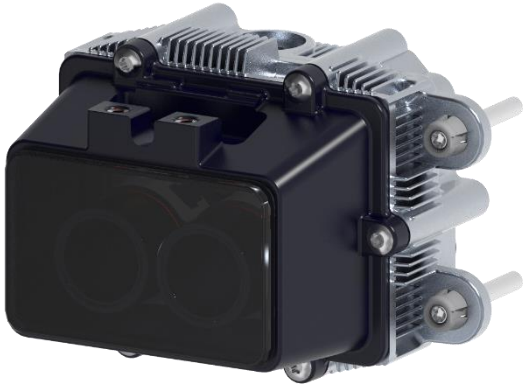
- Lidar
  - Illuminates a target with a laser and measures the characteristics of the reflected return signal.
    - Also called laser scanner, laser radar
  - Range: 0 – 200m
  - Pros: High resolution 3-D mapping, identify and classify objects
  - Cons: problems in rain, fog, snow; high cost
  - Mechanical Lidar: rotating assembly, 360° view, cost, size, robustness
    - “Orb” of google cars
  - Solid State Lidar: no spinning parts, multiple at front, rear and side combined together

https://www.ti.com/lit/wp/slyy150a/slyy150a.pdf?ts=1607565385866&ref_url=https%253A%252F%252Fwww.google.com%252F

Sensors & Applications: Autonomous Driving

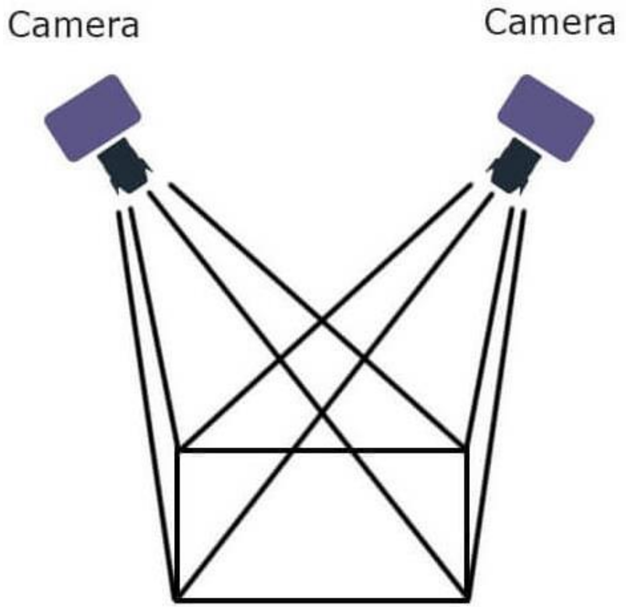
- Lidar
  - Mechanical Lidar: rotating assembly, 360° view, cost, size, robustness
    - “Orb” of google cars
  - Solid State Lidar: no spinning parts, multiple at front, rear and side combined together
    - Hundreds to thousands of lasers on each module
    - iPhone 12 Pro => better photos, 3D scanning, augmented reality





Sensors & Applications: Autonomous Driving

- Camera
  - Used to detect objects and classify objects
  - Mono camera: single camera
    - Object detection and classification
  - Stereo camera: two cameras at known offset
    - Spatial awareness: measure distance and improved size calculations
    - Pros: High resolution 3-D mapping, identify and classify objects
    - Cons: High cost, high computing and software requirements



FRSEF

Putting It All Together

FRSEF

- What are we trying to measuring? (our outcome)
  - Are there multiple ways we can measure it?
- What affects it? (our variables; independent, dependent and controlled)
  - Can we measure this?
- How fast will we measure and record it?
  - Are we limited by our equipment?
- How will we analyze it?
- How will we present our data?

Putting It All Together – Improved Helmet
FRSEF

- Engineering Goal: *Design a Helmet to reduce concussions in football.*
- What are we trying to measuring? *Impact to brain.*
  - *G Force's via accelerometer*
- What affects it? (our variables; independent, dependent and controlled)
  - Helmet Design (padding, shape, etc)
  - Drop Height
  - Environmental Factors? (temp, etc....)
  - Temp of padding, etc?
- How fast will we measure and record it?
- How will we analyze it?
- How will we present our data?

Putting It All Together – Best Fertilizer

FRSEF

- Hypothesis: *Scott's Fertilizer will produce the highest yield of tomatoes.*
- What are we trying to measuring? Yield of tomatoes – quantity, mass
- What affects it? (our variables; independent, dependent and controlled)
  - Fertilizer brand
  - Amount of applied fertilizer
  - Environmental Factors (temp, sunlight, soil moisture, water quantity)
  - Soil pH
- How fast will we measure and record it?
- How will we analyze it?
- How will we present our data?

Upcoming Activities

FRSEF

- Crash Course: Data Analysis
  - February
- 2021 Virtual Science Fair
  - Registering
    - www.flintsciencefair.org
  - Format, Resources and Templates
    - https://www.flintsciencefair.org/important-stuff/virtual-fair-information/
  - Senior Fair (9-12)
    - March 7: Registration Deadline + Upload of Project Materials
    - March 20: Judging Interviews (online via Zoom or similar)
  - Junior Fair (6-8) + Elementary Fair (4-5)
    - April 3: Registration Deadline + Upload of Project Materials
    - April 17: Judging Interviews (online via Zoom or similar)

Why Participate?

FRSEF

- Open to science, engineering, math and computer science projects
- Great learning experience
- Interact and communicate with local professionals
- Prizes!
  - Over $10,000 in cash prizes
  - Scholarships to Kettering and UM-Flint
  - 4 students advance to International Science and Engineering Fair (Senior category)
  - 15-20 students advance to Broadcom MASTERS (Junior category)

Resources

FRSEF

- Interactive Project Guide
  - Part 1: Starting a Project
  - Part 2: Experimentation and Communicating Results
- Educator Grants
- Student Project Grants

Lesson 7: Timing

Code Analysis: A new timing method

Lesson 7: Timing

```
const unsigned int interval = 1000;
static unsigned long nextTime = 0;
unsigned long time = millis();
if (time >= nextTime)
{
  // your code here
  nextTime = time + interval;
}
```

- The above code shows a better way to keep more accurate timing of code execution
- It uses the `millis()` function to keep track of the actual elapsed time and calculates when it needs to run the code again.
- It is different from the `delay()` function because `delay()` waits for a period of time whereas using the `millis()` method can compensate for the time it takes to execute code.
- See timing demos for an example of the running differences.

Activities

Lesson 7: Timing

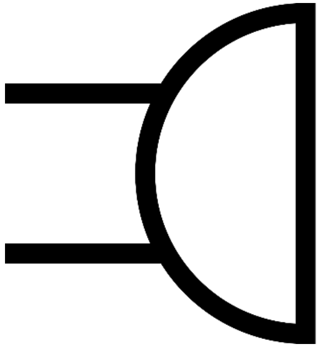
- Upload Sketch 7a and copy the output
- Upload Sketch 7b and copy the output
- Compare the outputs, which would you want to use for an application that requires precise timing?

Sensors & Applications

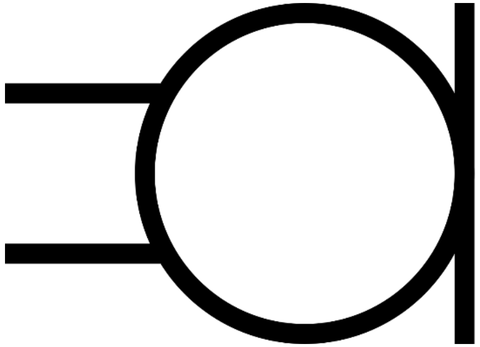
Sensors & Applications: Microphone

- What is a microphone?
  - A microphone is a transducer that converts sound wave to an electrical signal.
  - Microphones are used to record music and voice, but also used for scientific analysis.
- Where are microphones used?
  - Audio recording, cell phones, walkie-talkie, computers, sonar, presence detection, knock detection, etc.
- How do we use the Microphone?
  - Microphones must be amplified or conditioned before we can use the signal. We can then read the analog signal with the ADC in the microcontroller.
- More Information:
  - https://en.wikipedia.org/wiki/Microphone

Microphone Electrical Symbols



By ErikBuer - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=50257723



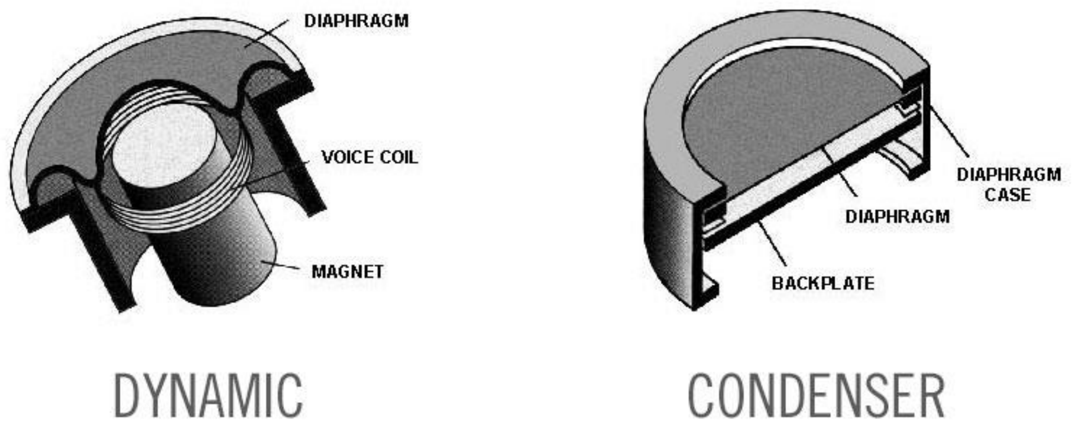
By ErikBuer - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=50257685

Microphone

Downloaded from https://www.digikey.com/en/products/detail/cui-devices/CMA-6542PF/1869980

Sensors & Applications: Microphone

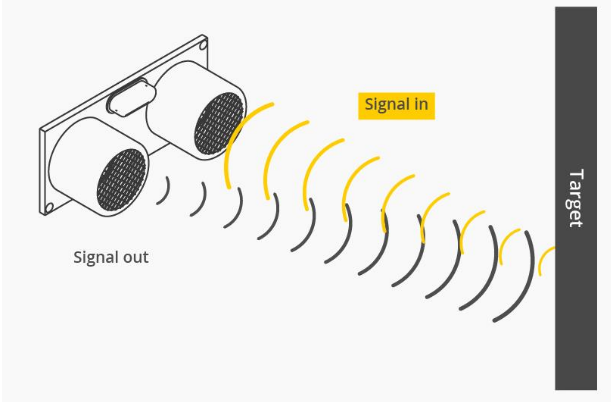
- There are multiple types of microphones, most common are dynamic and condenser



- More Information:
  - https://taylor-sound.com/taylor-sound-blog/whats-the-difference-between-a-dynamic-and-condenser-microphone/

Sensors & Applications: Ultrasonic Sensors

- Application using microphones: ultrasonic sensors
  - Ultrasonic Sensors
    - Simple range finders
    - Object position and tracking
      - Example: automotive active safety



- https://www.seeedstudio.com/blog/2020/01/03/what-is-a-sound-sensor-uses-arduino-guide-projects/#:~:text=A%20sound%20sensor%20is%20defined,converting%20it%20to%20electrical%20signals

Sensors & Applications: Ultrasonic Sensors

– Ultrasonic Sensors
  - Object position and tracking
    – Example: automotive active safety