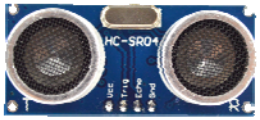
IOT DEVICES AND PLATFORMS

LAST WEEK
ARDUINO BUILT-IN EXAMPLE: COMMUNICATION

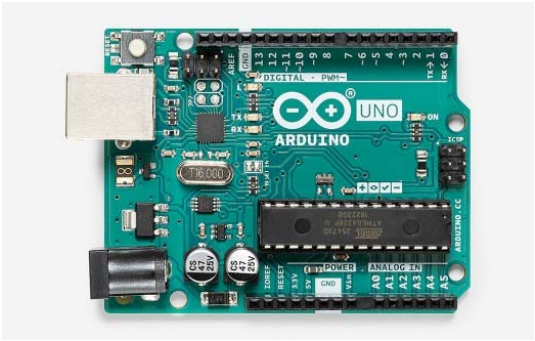
1. **ReadASCIIString:** Parse a comma-separated string of ints to fade an LED.
2. **ASCIITable:** Demonstrates Arduino's advanced serial output functions.
3. **Dimmer:** Move the mouse to change the brightness of an LED.
4. **Graph:** Send data to the computer and graph it in Processing.

5269 物聯網裝置與平台
IOE5114 IOT DEVICES AND PLATFORMS

- Course Outline
  - Chapter 0: Class rule
  - Chapter 1: IOT devices and sensors introduction
  - Chapter 2: Arduino built-in example: Basics
  - Chapter 3: Arduino built-in example: Digital
  - Chapter 4: Arduino built-in example: Analog
  - Chapter 5: Arduino built-in example: Communication
  - Chapter 6: Sensor application
  - Chapter 7: Arduino wireless networking
  - Chapter 8: IOT Cloud platform
  - Chapter 9: AI application

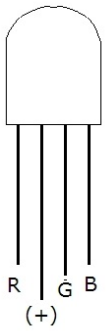

CH. 6 SENSOR APPLICATION (PART 2)



4

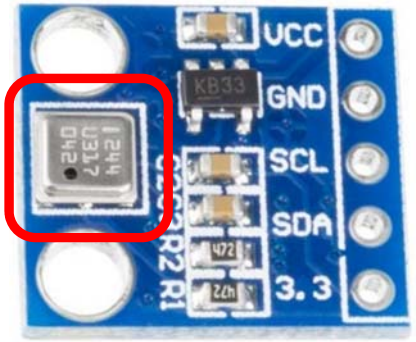
TUTORIALS

1. **Sweep:** Sweeps the shaft of a RC servo motor (Radio Control or Remote control) back and forth across 180 degrees.
2. **Knob:** Control the position of a RC servo motor with Arduino and a potentiometer.
3. **Altimeter:** Use BMP180 to read the pressure and temperature, and then calculate the altitude.



RGB Led

servo motor

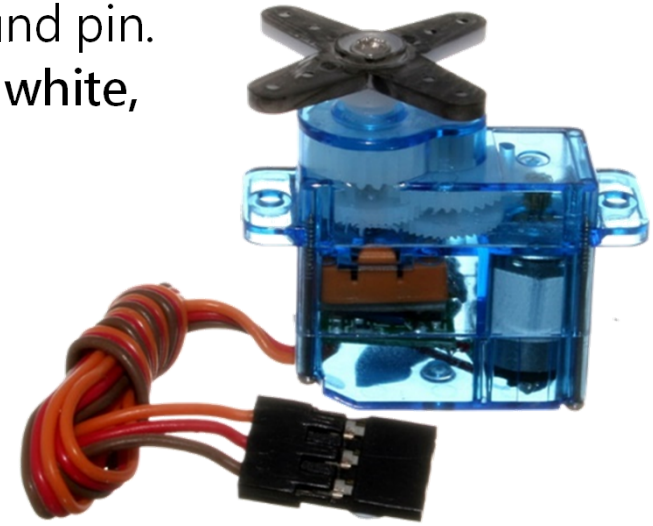


Pressure (altimeter)

RC SERVO

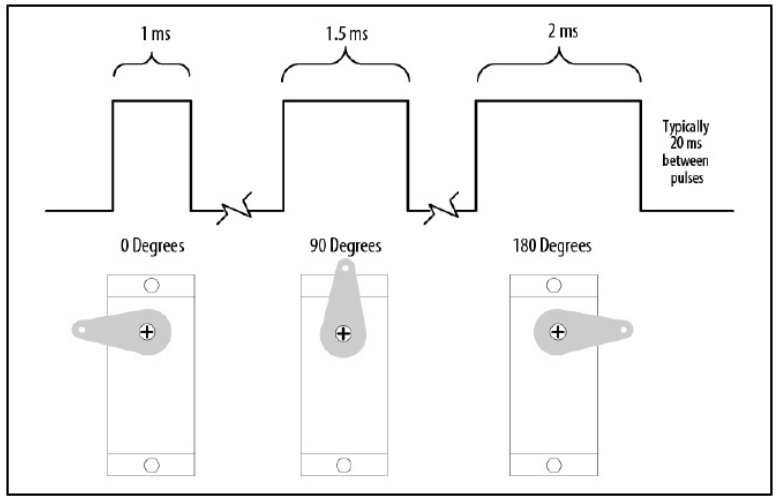
- Servos are small, cheap, mass-produced actuators used for **radio control** and **small-scale robotics**.
- Servo motors have three wires: power, **ground**, and signal.
  - Power: typically **red**, connect to the 5V pin.
  - Ground: typically **black** or **brown,** connect to a ground pin.
  - Signal: the remaining pin, maybe **yellow**, **orange** or **white,** connect to digital pin.

three wires:
power, **ground**, and **signal**



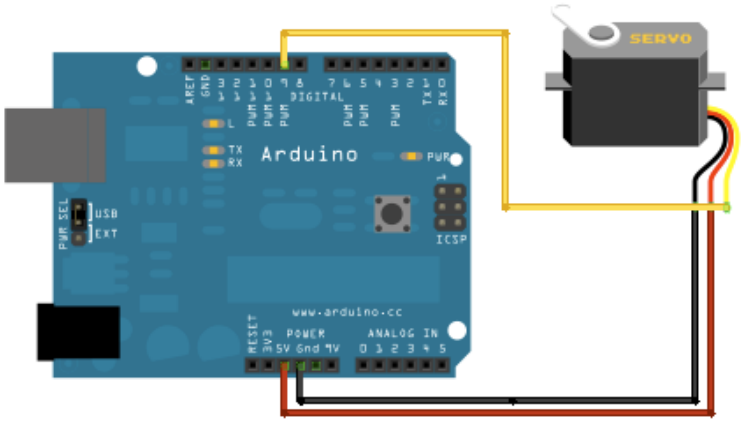
RC SERVO (CONT.)

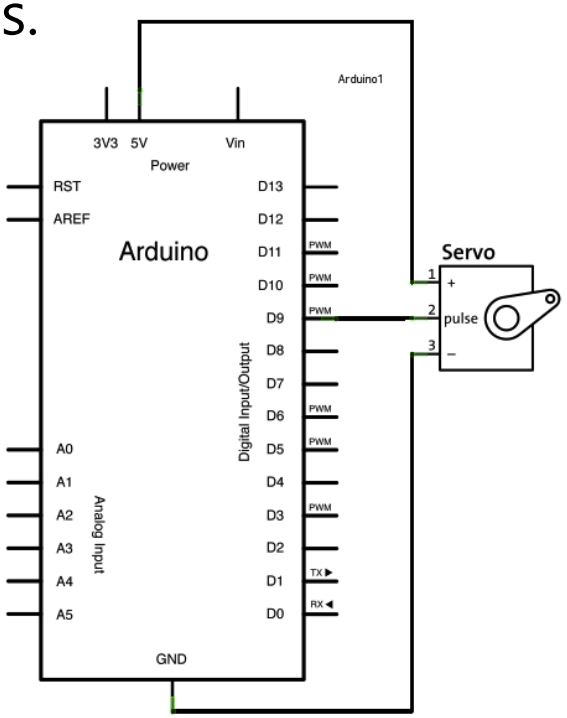
- The signal pin accepts the control signal which is a pulse-width modulation (**PWM**) signal.
- (Example) The pulse width sent to servo ranges as follows:
  - **Minimum**: 1 ms ---> Corresponds to 0 rotation angle.
  - **Maximum**: 2 ms ---> Corresponds to 180 rotation angle.
  - Please refer to its spec..



TUT. 1. SWEEP

- Goal: Sweeps the shaft of a RC (**Radio Control** or **Remote control**) servo motor back and forth across 180 degrees.
- Hardware Required
  - Arduino
  - Servo Motor
  - Hook-up wire

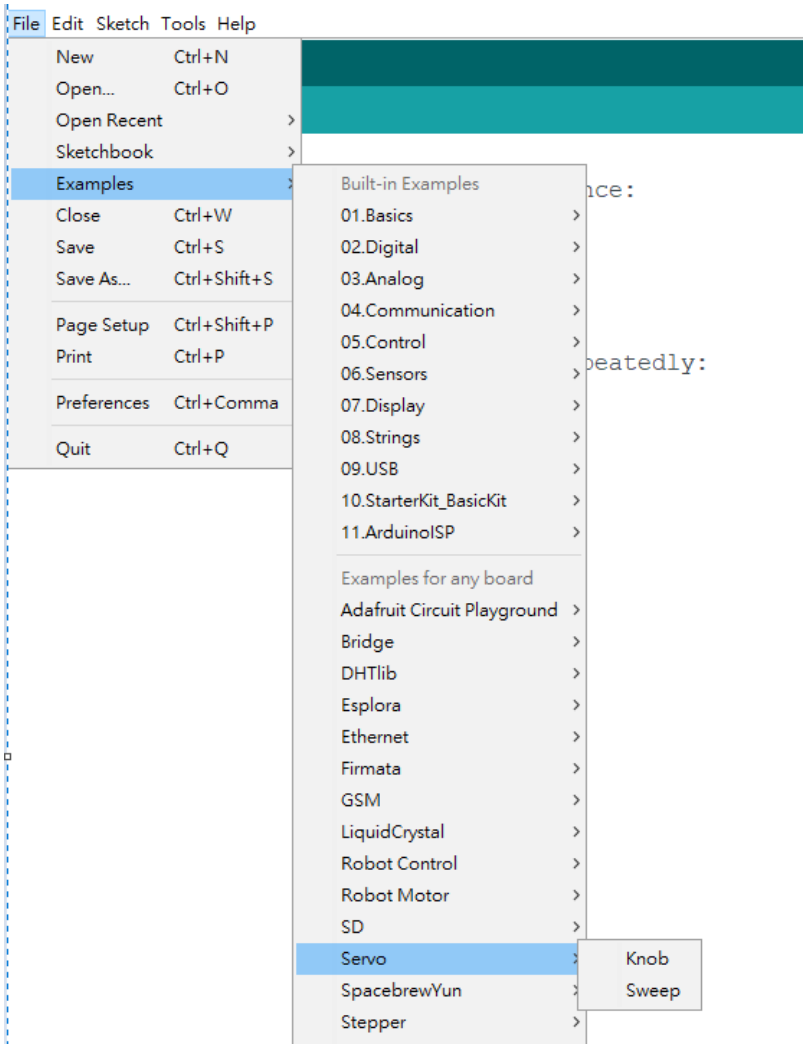




https://www.arduino.cc/en/tutorial/sweep

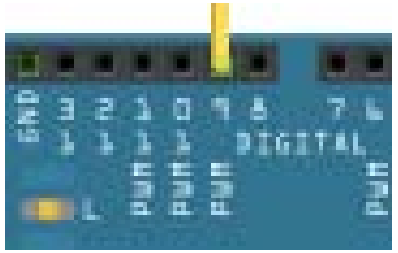
TUT. 1. SWEEP (CONT.)

- Arduino
- **Open--->File--->Examples--->Servo-->Sweep**



TUT. 1. SWEEP (CONT.)

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```



myservo.write(0) -> myservo.write(1) -> … -> myservo.write(179) -> myservo.write(180)
-> myservo.write(179) -> myservo.write(178) -> … -> myservo.write(0)

TUT. 1. SWEEP (CONT.)

- Syntax
  - servo.attach(**pin**)
  - servo.attach(**pin**, min, max)

- Parameters
  - **pin**: the **pin number** that the servo is attached to (usually, Pin 9 or 10)
  - **min (optional):** the pulse width, in microseconds, corresponding to the minimum (0-degree) angle on the servo (defaults to 544)
  - **max (optional):** the pulse width, in microseconds, corresponding to the maximum (180-degree) angle on the servo (defaults to 2400)

https://www.arduino.cc/en/Reference/ServoAttach

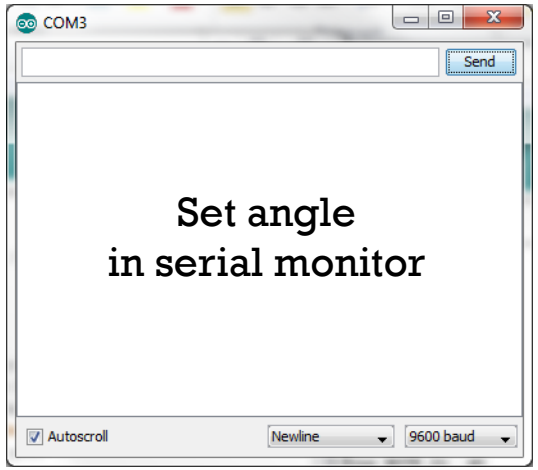
TUT. 1. SWEEP (CONT.)

- Syntax
  - servo.write(angle)
- Description
  - Writes a value to the servo, controlling the shaft accordingly.
  - On a **standard servo**, this will set **the angle of the shaft (in degrees)**, moving the shaft to that orientation.
  - On a continuous rotation servo, this will set the speed of the servo (with 0 being full-speed in one direction, 180 being full speed in the other, and a value near 90 being no movement).
- Parameters
  - angle: the value to write to the servo, from **0 to 180** (in degree)
- Example
  - myservo.write(90); *// set servo to mid-point*

https://www.arduino.cc/en/Reference/ServoWrite

LAB: DEMO 1

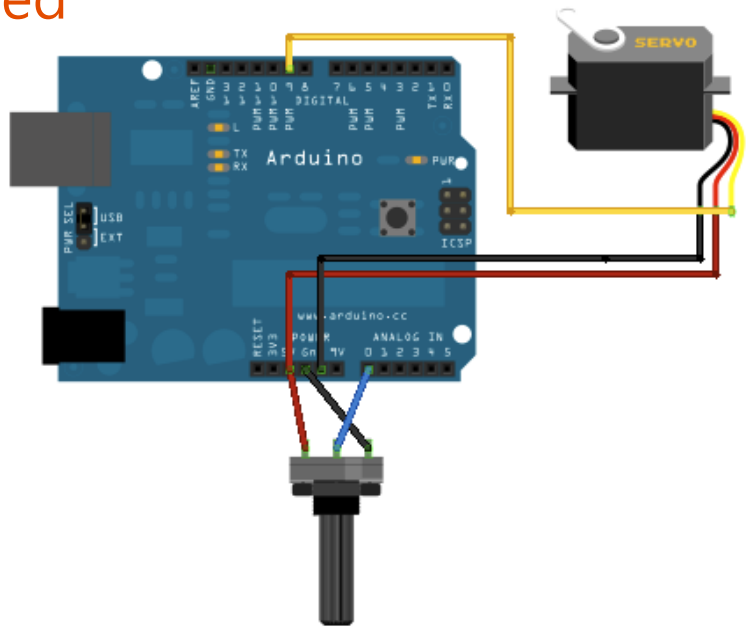
- In lab 1, RC servo motor sweeps continuously.
- How do we set a specific angle for RC motor in serial monitor?
  - The servo motor does not sweep continuously
  - Hint: we introduce communication last week

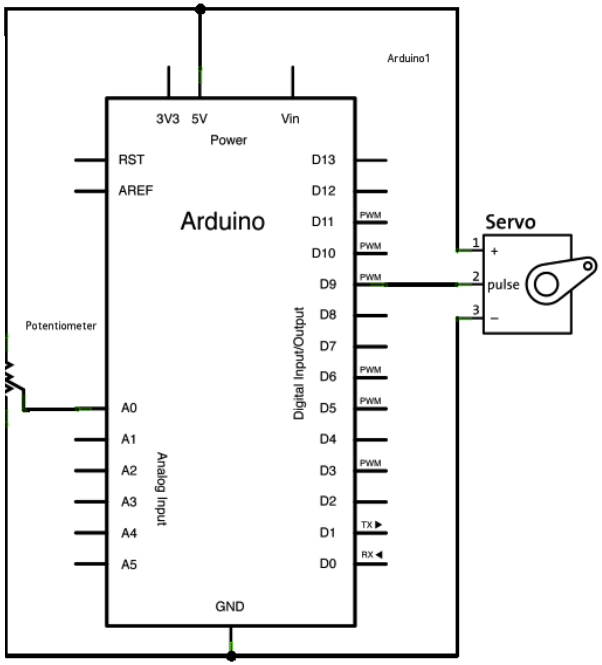


Control the angle of the shaft

TUT. 2. KNOB

- Goal: Control the position of a RC servo motor with your Arduino and a potentiometer.
- Hardware Required
  - Arduino
  - Servo Motor
  - Potentiometer
  - Hook-up wire



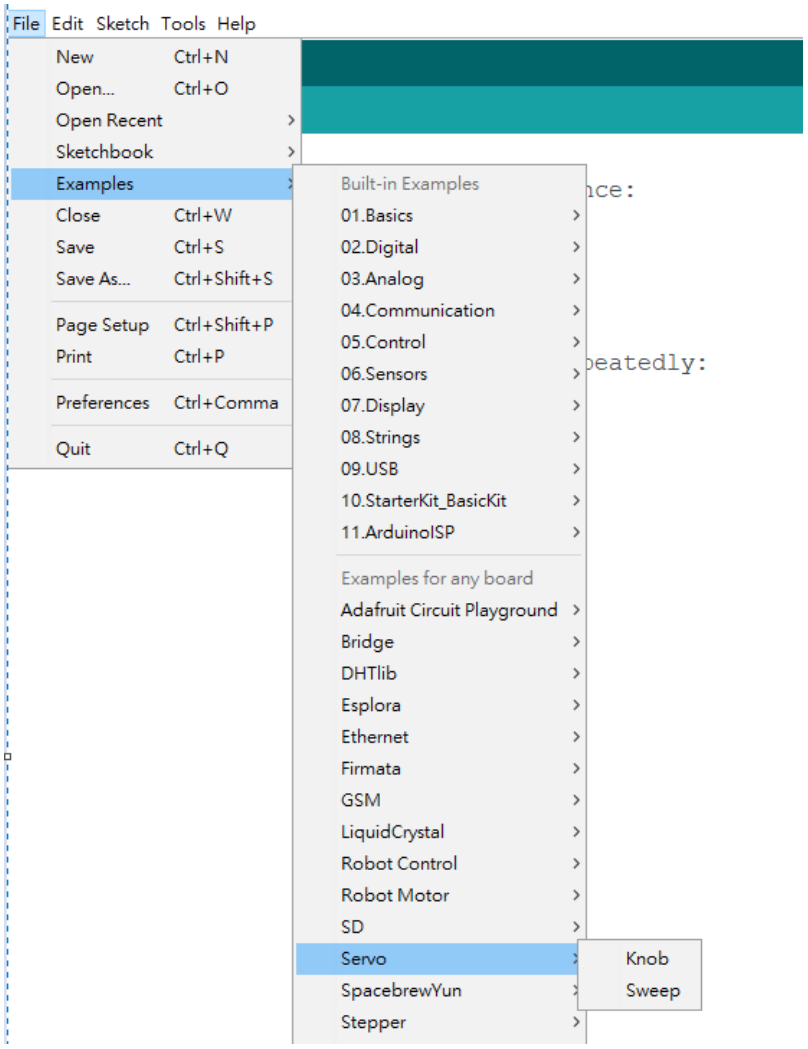


https://www.arduino.cc/en/tutorial/knob

TUT. 2. KNOB (CONT.)

Arduino

- Open--->File--->Examples--->Servo-->Knob



TUT. 2. KNOB (CONT.)

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo

int potpin = 0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  val = analogRead(potpin);            // reads the value of the potentiometer (value between 0 and 1023)
  val = map(val, 0, 1023, 0, 180);     // scale it to use it with the servo (value between 0 and 180)
  myservo.write(val);                  // sets the servo position according to the scaled value
  delay(15);                           // waits for the servo to get there
}
```

TUT. 2. KNOB (CONT.)

- analogRead
  - 10-bit analog to digital converter
  - https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/

- map
  - Re-maps a number from one range to another.
  - https://www.arduino.cc/reference/en/language/functions/math/map/

- servo write
  - Writes a value to the servo, controlling the shaft accordingly.
  - https://www.arduino.cc/en/Reference/ServoWrite

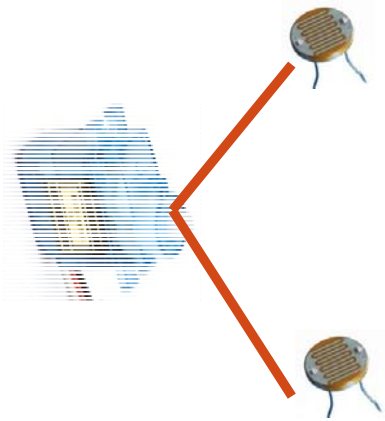
LAB: DISCUSSION 1

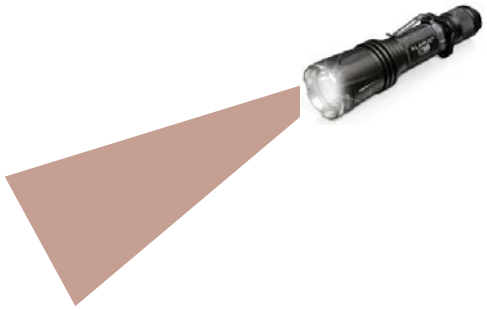
- After myservo.write(val), why do we need delay(15)? What will happen if we don't use delay?

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo

int potpin = 0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  val = analogRead(potpin);            // reads the value of the potentiometer (value between 0 and 1023)
  val = map(val, 0, 1023, 0, 180);     // scale it to use it with the servo (value between 0 and 180)
  myservo.write(val);                  // sets the servo position according to the scaled value
  delay(15);                           // waits for the servo to get there
}
```

LAB: DEMO 2

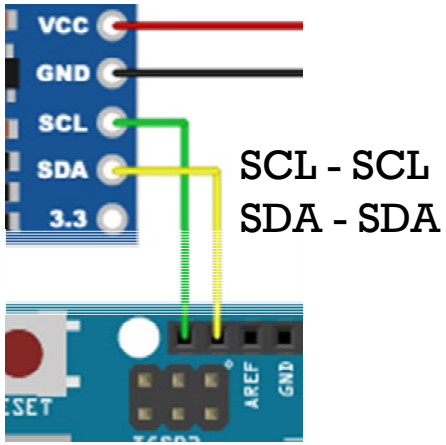
- Design a sunflower by photocells, servo motor
  - Attach photocell to servo
  - When one moves the light source (e.g., torch from smartphone), the servo moves the photocells till the photocells face the light source.
  - Sweep the photocell to find the position with maximum illumination



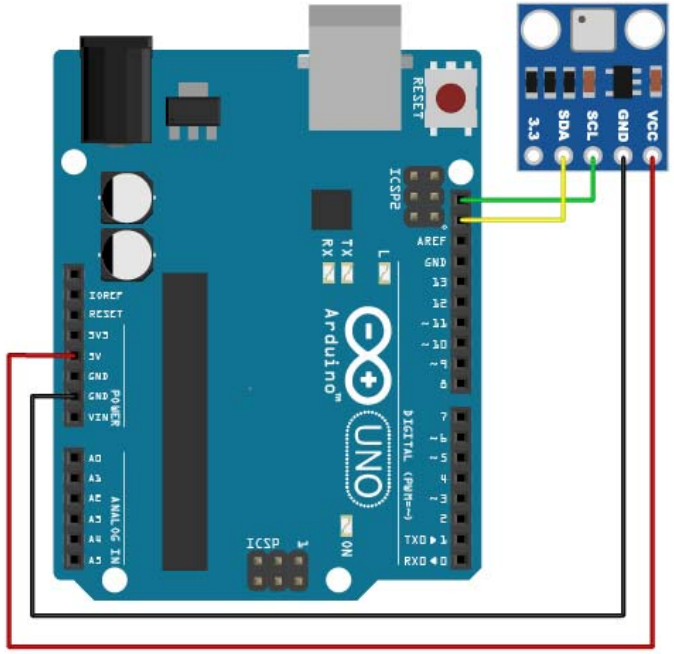


TUT. 3. ALTIMETER

- Goal: Use BMP180 to read the pressure and temperature, and then calculate the altitude and floor number.
- Hardware Required
  - Arduino
  - BMP180



SCL - SCL
SDA - SDA



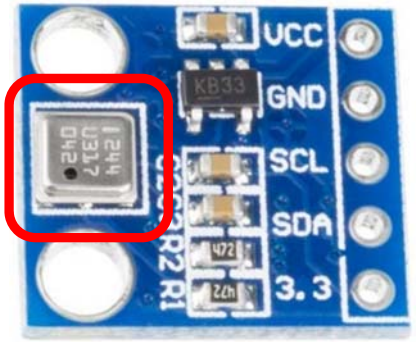
https://github.com/jrowberg/i2cdevlib

TUT. 3. ALTIMETER (CONT.)

- BMP180: measure barometric pressure and temperature.
  - The pressure changes with altitude, we can also use it as an altimeter.
  - Use I2C-bus to read the sensor values.

Technical Details

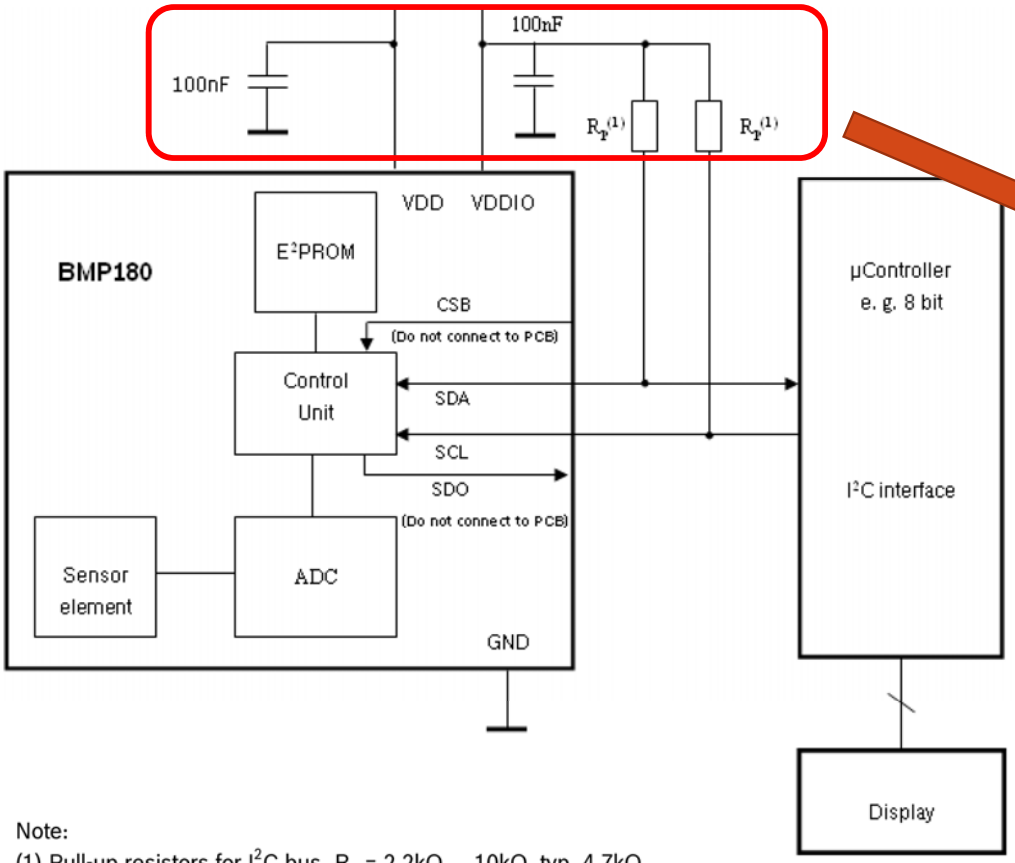
- VCC: 3 to 5V
- Pressure sensing range: 300-1100hPa
- 9000m to -500m above sea level
- Up to 0.03hPa / 0.25m resolution
- Temperature sensing range: -40 to 85˚C
- +-2˚C temperature accuracy
- Use I2C address 0x77

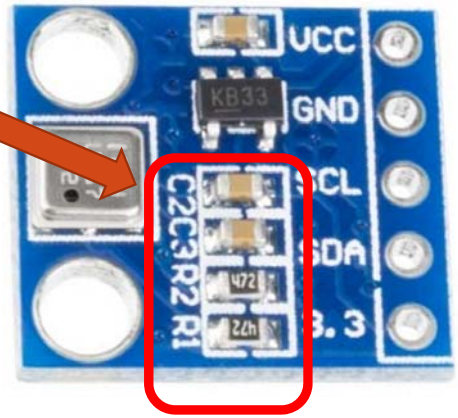


https://static.sparkfun.com/datasheets/Sensors/Pressure/BMP180.pdf

https://learn.sparkfun.com/tutorials/bmp180-barometric-pressure-sensor-hookup-

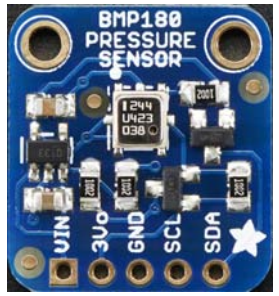
TUT. 3. ALTIMETER (CONT.)





Note:
(1) Pull-up resistors for I^2C bus, R_p = 2.2kΩ ... 10kΩ, typ. 4.7kΩ

TUT. 3. ALTIMETER (CONT.)



- Two common model: BMP 085 and BMP 180

- What is the difference?
  - The BMP180 is the next-generation of sensors from Bosch, and replaces the BMP085.
  - BMP180 is **completely identical** to the BMP085 **in terms of firmware/software**

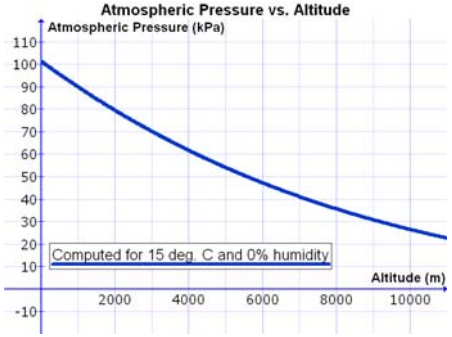
https://www.adafruit.com/product/1603

TUT. 3. ALTIMETER (CONT.) PRESSURE AND ALTITUDE

- Barometric formula (valid within troposphere) :

$$p = p_0 \cdot \left(1 - \frac{L \cdot h}{T_0}\right)^{\frac{g \cdot M}{R_0 \cdot L}} \approx p_0 \cdot \left(1 - \frac{g \cdot h}{c_p \cdot T_0}\right)^{\frac{c_p \cdot M}{R_0}},$$

$$p \approx p_0 \cdot \exp\left(-\frac{g \cdot M \cdot h}{R_0 \cdot T_0}\right)$$



Parameter	Description	Value
p_0	sea level standard atmospheric pressure	101325 Pa
L	temperature lapse rate, = g/c_p for dry air	0.0065 K/m
c_p	constant pressure specific heat	~ 1007 J/(kg•K)
T_0	sea level standard temperature	288.15 K
g	Earth-surface gravitational acceleration	9.80665 m/s^2
M	molar mass of dry air	0.0289644 kg/mol
R_0	universal gas constant	8.31447 J/(mol•K)

https://en.wikipedia.org/wiki/Atmospheric_pressure

TUT. 3. ALTIMETER (CONT.) PRESSURE AND ALTITUDE

- How to calculate altitude?
  1. Start temperature measurement
  2. Start pressure measurement
  3. Based on sea-level and measured pressure, calculate the altitude

$$p = p_0 \cdot \left(1 - \frac{L \cdot h}{T_0}\right)^{\frac{g \cdot M}{R_0 \cdot L}} \approx p_0 \cdot \left(1 - \frac{g \cdot h}{c_p \cdot T_0}\right)^{\frac{c_p \cdot M}{R_0}}$$

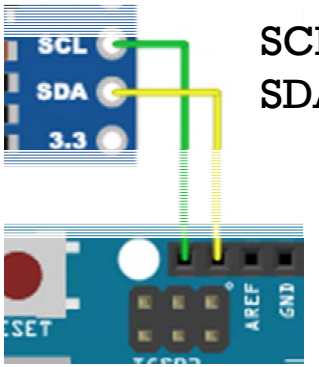
$$p \approx p_0 \cdot \exp\left(-\frac{g \cdot M \cdot h}{R_0 \cdot T_0}\right)$$

$$\text{altitude} = 44330 * \left(1 - \left(\frac{p}{p_0}\right)^{\frac{1}{5.255}}\right)$$

BMP085.cpp

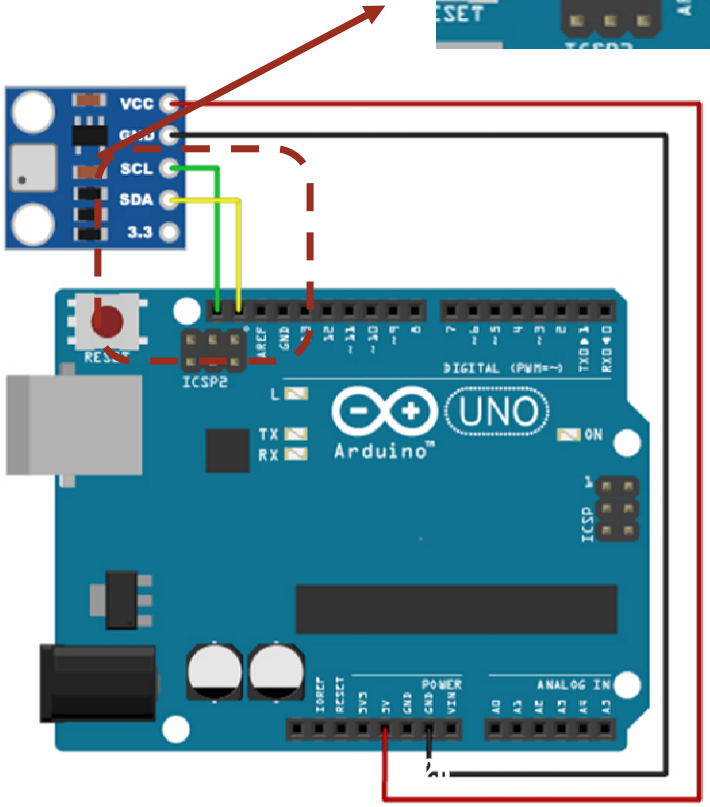
```
float BMP085::getAltitude(float pressure, float seaLevelPressure)
{    return 44330 * (1.0 - pow(pressure / seaLevelPressure, 0.1903));}
```

TUT. 3. ALTIMETER (CONT.)



SCL - SCL
SDA - SDA

Arduino

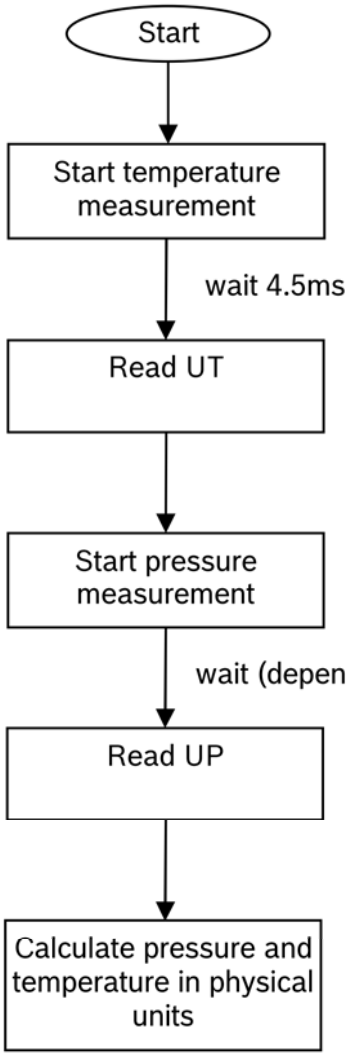


- Download the library and code

BMP085.cpp
BMP085.h
BMP085_basic
I2Cdev.cpp
I2Cdev.h

TUT. 3. ALTIMETER (CONT.)

```
#include "Wire.h "
#include "I2Cdev.h "
#include "BMP085.h "
BMP085 barometer;

float temperature;
float pressure;
float seaLevelPressure = 101325;
float altitude;
int32_t lastMicros;

#define LED_PIN 13
bool blinkState = false;
```



The standard atmosphere (1 atm) is a unit of pressure defined as 101325 Pa.

TUT. 3. ALTIMETER (CONT.)

```
void setup() {
    Wire.begin();
    Serial.begin(38400);
    // initialize device
    Serial.println("Initializing I2C devices...");
    barometer.initialize();
    // verify connection
    Serial.println("Testing device connections...");
    Serial.println(barometer.testConnection() ? "BMP085 connection successful" :
"BMP085 connection failed");
    // configure LED pin for activity indication
    pinMode(LED_PIN, OUTPUT);
}
```

TUT. 3. ALTIMETER (CONT.)

```
void loop() {
   // request temperature
   barometer.setControl(BMP085_MODE_TEMPERATURE);
   // wait appropriate time for conversion (4.5ms delay by datasheet)
   lastMicros = micros();

   while (micros() - lastMicros < barometer.getMeasureDelayMicroseconds());

   // read calibrated temperature value in degrees Celsius
   temperature = barometer.getTemperatureC();

   // Next
```

TUT. 3. ALTIMETER (CONT.)

```
// Cont.
    // request pressure
barometer.setControl(BMP085_MODE_PRESSURE_3);

while (micros() - lastMicros < barometer.getMeasureDelayMicroseconds());

// read calibrated pressure value in Pascals (Pa)
pressure = barometer.getPressure();

// calculate absolute altitude in meters based on known pressure
altitude = barometer.getAltitude(pressure, seaLevelPressure);

// Next
```

TUT. 3. ALTIMETER (CONT.)

```
// display measured values if appropriate
   Serial.print("T/P/A\t");
   Serial.print(temperature);          Serial.print("\t");
   Serial.print(pressure);             Serial.print("\t");
   Serial.print(altitude);             Serial.println("");

   // blink LED to indicate activity
   blinkState = !blinkState;
   digitalWrite(LED_PIN, blinkState);

   // delay 100 ms to allow visually parsing blink and any serial output
   delay(100);
}
```

LAB: DISCUSSION 2

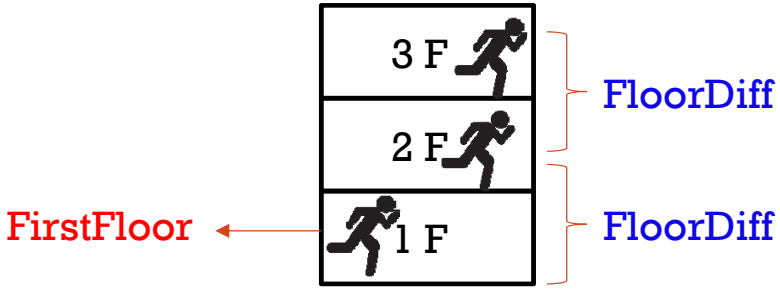
1. Study the code in **BMP085.cpp**. How does the altitude is calculated by using pressure?
2. What is **the unit** of the altitude formula? (as follows)
3. Try to lift your BMP180 sensor, and **write down the value change** between put it on the table and lift it up.

$$\text{altitude} = 44330 * \left(1 - \left(\frac{p}{p_0}\right)^{\frac{1}{5.255}}\right)$$ **(unit ???)**

LAB: DEMO 3

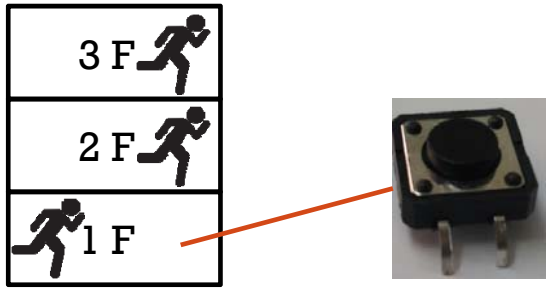
- Use the following formula to design a machine to **detect the floor number** of Engineer Building 3
  - Use one button and three LED

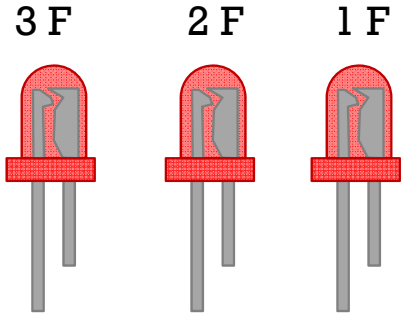
- $Floor = \frac{Current\_Altitude - FirstFloor}{FloorDiff} + 1$
  - FirstFloor: the altitude of the first floor
  - FloorDiff: the altitude difference between floors



LAB: DEMO 3 (CONT.)

- Steps:
  1. Press a button to measure the altitude of first floor
  2. Press a button to measure the altitude of second floor and calculate the altitude difference between the two floors
  3. Press button again and use 3 LEDs to present the floor number





Move with your laptop or power bank

SUMMARY

- Practice TUT. 1 to 3

- Write answer for Discussion 1 to 2
  - Upload to new e3 before 11/2 (next class) (No plagiarizing)
  - **If you upload within one week after deadline** (After that we do not accept your work.) or **if you do not follow the format specified by TAs**, your score will be multiplied by 0.8.
- Write code for Demo 1 to 3, then demonstrate it to TAs
  - If you can not finish today, remember to tell TA before you leave. (record your attendance)
  - If you demonstrate quiz in next week, the score is multiply by 0.8 (ex: 100 -> 80)

NEXT WEEK

- Use I2C-bus or a single data line to read the sensor values.

1. **Accelerometer + Gyroscope** (MPU-6050)
   http://playground.arduino.cc/Main/MPU-6050
2. **Magnetometer (Compass)** (HMC-5883L)
   http://playground.arduino.cc/Main/InterfacingWithHardware

Accelerometer + gyroscope

Magnetometer