**Processors:**

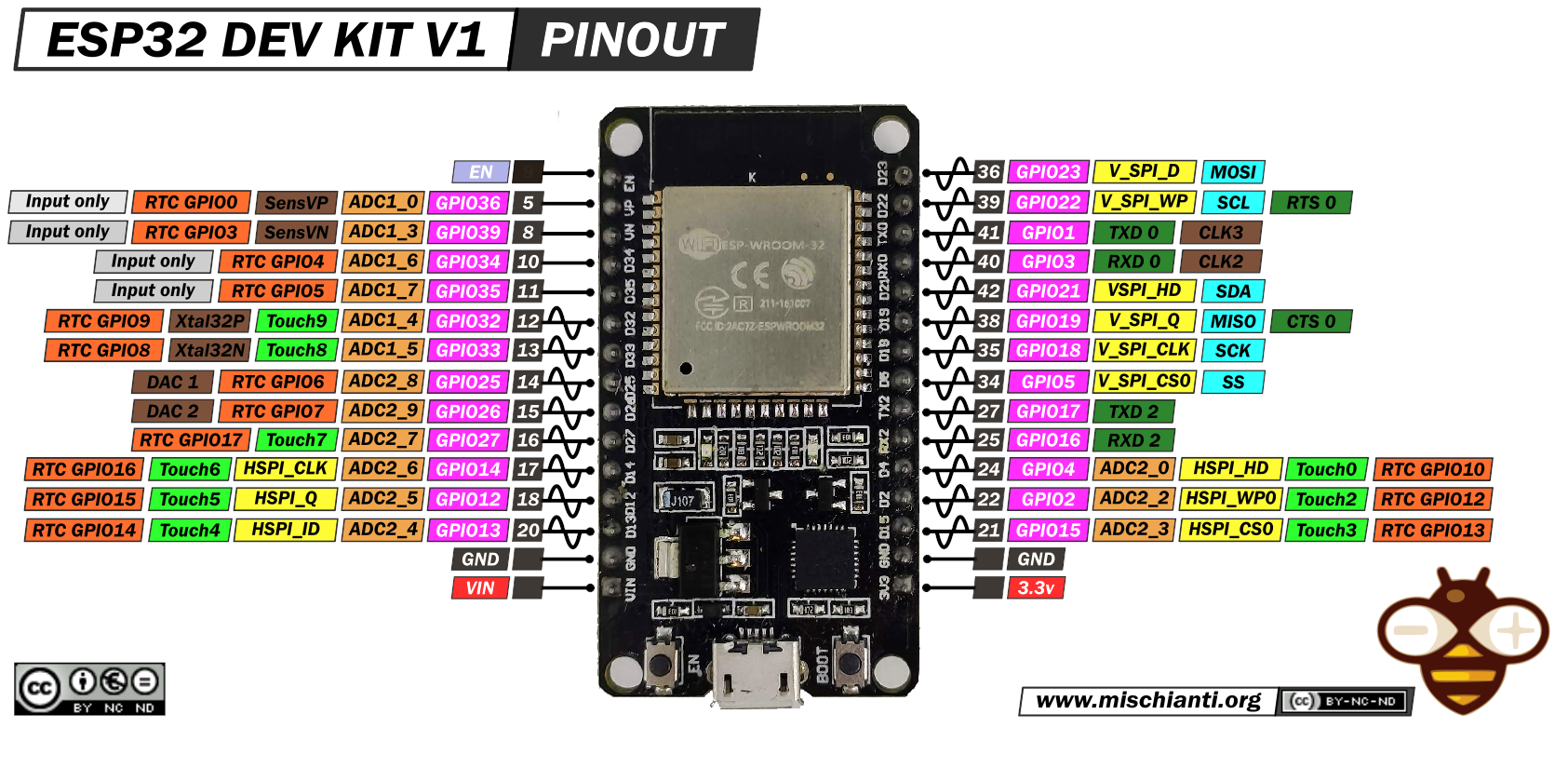
* Arduino Uno R3
* ESP 8266
* ESP 32

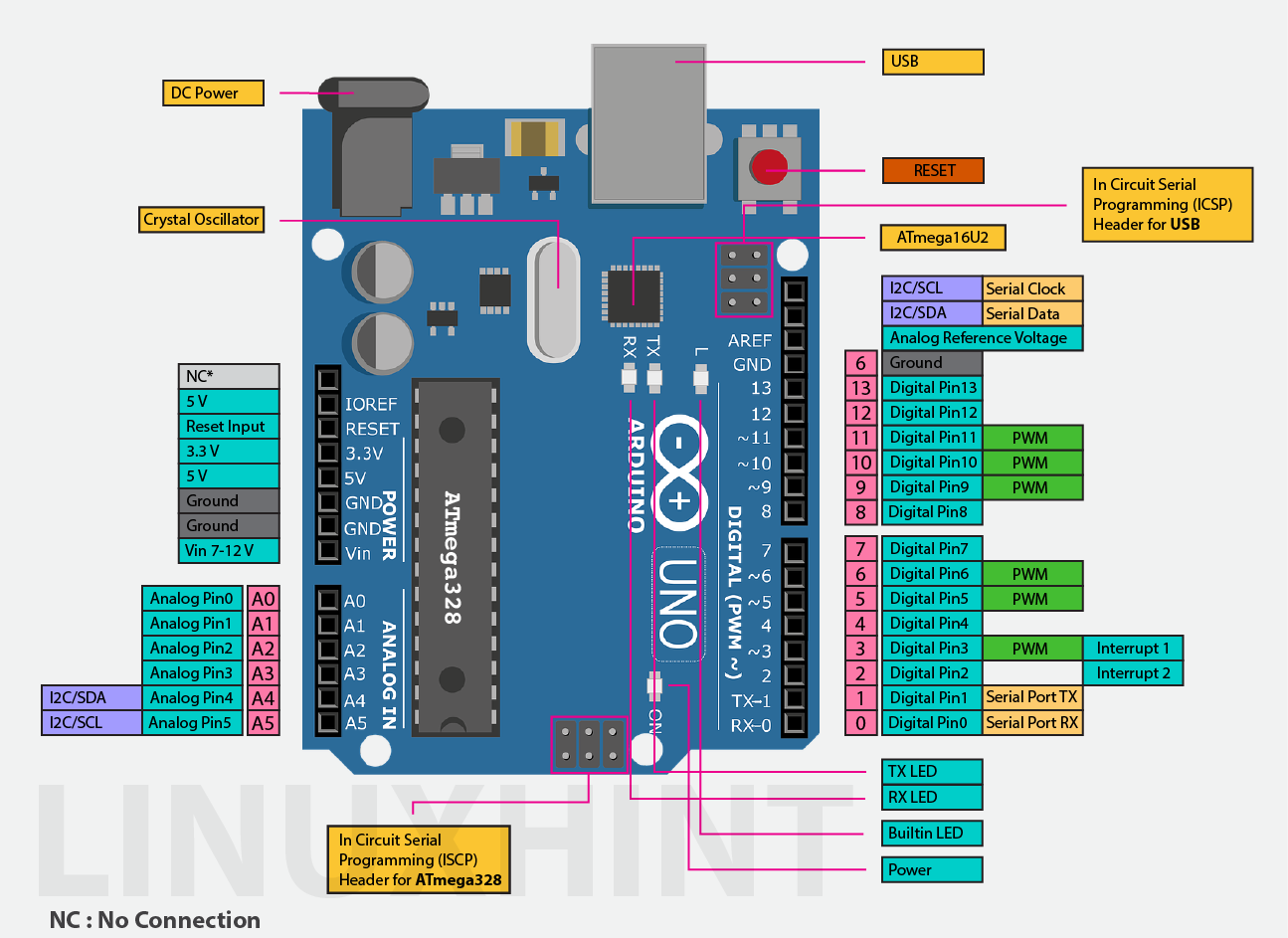
**Battery & Display**

* Pako 6F22 (**9 volt**)
* Qapass LCD Display (16X2)

**Sensor devices:**

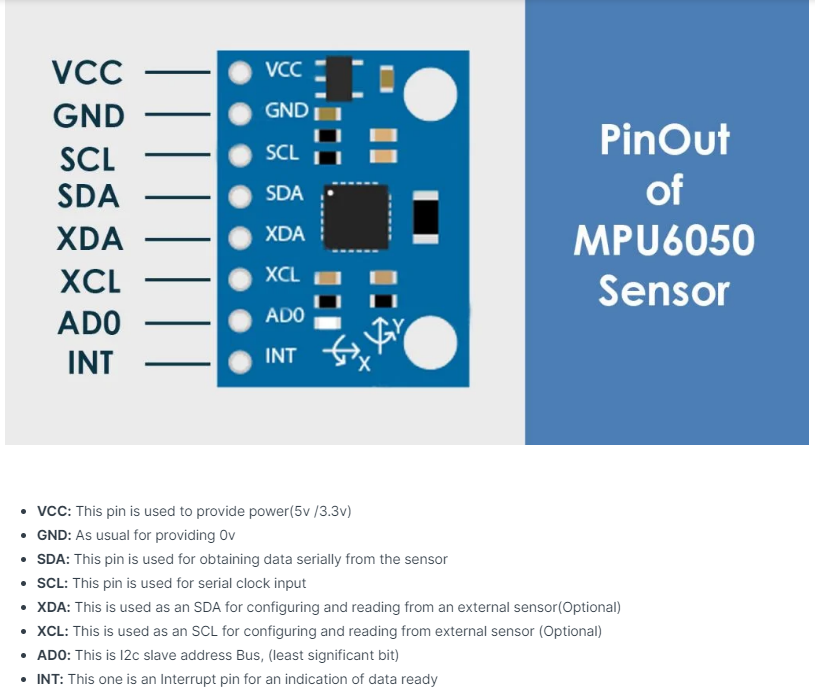
* MPU 6050
  + Accelerometer
  + Gyroscope
  + Gravity
  + Temperature
* PIR Motion
  + Motion
* Ultrasonic sensor HC-SR04
  + Distance
  + Proximity
* MQ 135
  + Gas (co2, co, methane)
* BMP 180
  + Temperature
  + Altitude
  + Pressure
* Humidity
  + Humidity
* GY30102 (MAX30102)
  + Pulse oximeter
  + Heart rate





**MPU 6050 MODULE**

|  |  |
| --- | --- |
| PIN | Link with Arduino |
| VCC - voltage | Works with both 3.3 or 5.0 volt |
| GND - ground | Connect with ground |
| SCL – serial clock line | Gpio18 or A5 |
| SDA – serial data line | Gpio19 or A4 |
| XCL – extra clock line |  |
| XDA – extra data line |  |
| INT – interrupt |  |
| AD0 – address bus |  |
| Last 4 pins are unnecessary and not even needed to mentioned in paper. | |



VCC: This pin is used to provide power (5v /3.3v)

GND: As usual for providing 0v

SDA: This pin is used for obtaining data serially from the sensor

SCL: This pin is used for serial clock input

XDA: This is used as an SDA for configuring and reading from an external sensor (Optional)

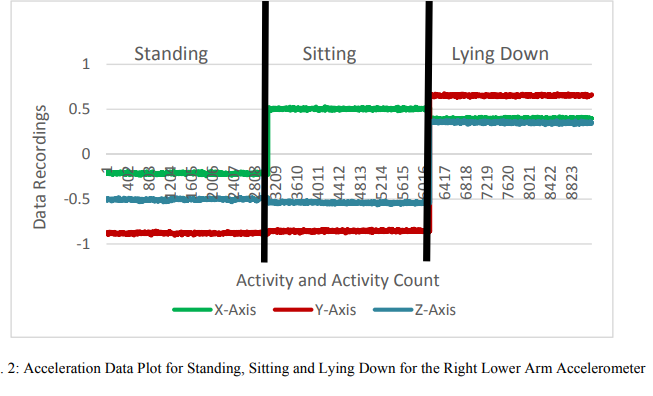
XCL: This is used as an SCL for configuring and reading from external sensor (Optional)

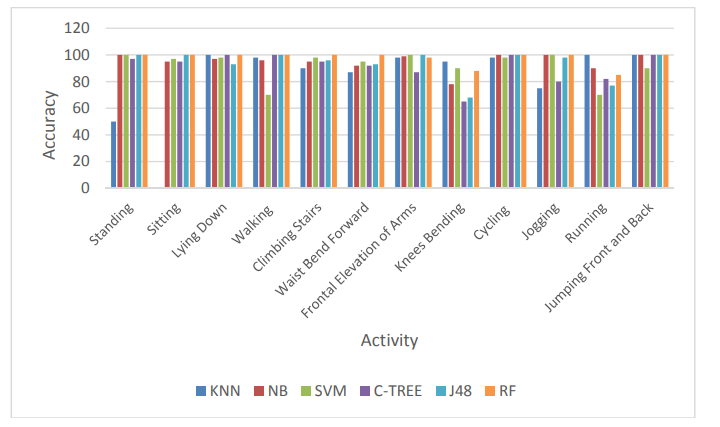
AD0: This is I2c slave address Bus, (least significant bit)

INT: This one is an Interrupt pin for an indication of data ready

***[helper paper – 1] Human Body and Body Part Movement Analysis Using Gyroscope, Accelerometer and Compass***

***[helper paper – 2] Human Activity Recognition Using Accelerometer and Gyroscope Sensors***





***[helper paper – 3] Reliability of Wireless Body Area Networks used for Ambulatory Monitoring and Health Care***

**Cases study:**

* **If we take accelerometer (x or y or z) reading directly under consideration than, the results will carry out nothing useful! Why? Because the kid will not hold the device/sensor always in same orientation. Also, he’ll be moving so X axis of sensor perhaps will read movements of Y or Y may read the Z’s. So direct reading can do nothing.**
* **Sensor gives both positive and negative results. But we don’t have that’s use as well. Because we only need to detect the movement. Not the direction of movement.**
* **So, converting all reading into absolute value will work fine.**
* **Now still, how can we differentiate normal readings and risky readings? Following table shows some experiments results -**

|  |  |
| --- | --- |
| Cases | Reading |
| Space | Acc total = 0 |
| Forced not to move | Acc total = ~1.1 |
| Under fan wind | Acc total = ~1.6 |
| Normal hand move | Acc total = ~2.0-2.6 |
| Tuka (fast) | Acc total = ~3.0-4.5 |
| Very fast shaking | Acc total = ~6.0 |

**So, normal movements don’t even close to falling (9.8m/s^2) acceleration reading.**

**Till now if (reading >= g) obviously risk**

**More than 2.6? === Maybe risk, but need other sensors help to make sure.**

**CODE**

#include <MPU6050\_tockn.h>

#include <Wire.h>

MPU6050 mpu6050(Wire);

long timer = 0;

void setup() {

Serial.begin(1000000);

Wire.begin();

mpu6050.begin();

mpu6050.calcGyroOffsets(true);

}

void loop() {

mpu6050.update();

if(millis() - timer > 200){

Serial.println("-------------------------------------");

Serial.print("temp : ");

Serial.println(mpu6050.getTemp());

Serial.print("accX : ");

Serial.print(mpu6050.getAccX());

Serial.print("\taccY : ");

Serial.print(mpu6050.getAccY());

Serial.print("\taccZ : ");

Serial.println(mpu6050.getAccZ());

Serial.print("gyroX : ");

Serial.print(mpu6050.getGyroX());

Serial.print("\tgyroY : ");

Serial.print(mpu6050.getGyroY());

Serial.print("\tgyroZ : ");

Serial.println(mpu6050.getGyroZ());

Serial.print("accAngleX : ");

Serial.print(mpu6050.getAccAngleX());

Serial.print("\taccAngleY : ");

Serial.println(mpu6050.getAccAngleY());

Serial.print("gyroAngleX : ");

Serial.print(mpu6050.getGyroAngleX());

Serial.print("\tgyroAngleY : ");

Serial.print(mpu6050.getGyroAngleY());

Serial.print("\tgyroAngleZ : ");

Serial.println(mpu6050.getGyroAngleZ());

Serial.print("angleX : ");

Serial.print(mpu6050.getAngleX());

Serial.print("\tangleY : ");

Serial.print(mpu6050.getAngleY());

Serial.print("\tangleZ : ");

Serial.println(mpu6050.getAngleZ());

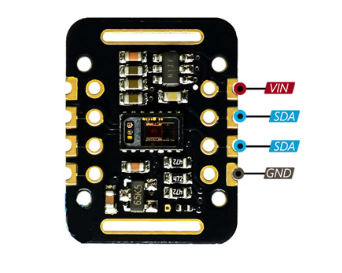
Serial.println("----------------------------------\n");

timer = millis();

}

}

**MAX30102 Heart Rate Module**



* **VCC:** Module power supply – 3 to 5 V
* **GND**: Ground
* **SCL**: I2C clock bus
* **SDA**: I2C data bus

Library: MX30 SparkFun

**CODE:**

#include <MPU6050\_tockn.h>

#include <Wire.h>

MPU6050 mpu6050(Wire);

long timer = 0;

void setup() {

Serial.begin(115200);

Wire.begin();

mpu6050.begin();

mpu6050.calcGyroOffsets(true);

}

void loop() {

motion();

}

void motion(){

mpu6050.update();

Serial.println("-------------------------------------");

Serial.print("temp : ");

Serial.println(mpu6050.getTemp());

Serial.print("accX : ");

Serial.print(mpu6050.getAccX());

Serial.print("\taccY : ");

Serial.print(mpu6050.getAccY());

Serial.print("\taccZ : ");

Serial.println(mpu6050.getAccZ());

Serial.print("gyroX : ");

Serial.print(mpu6050.getGyroX());

Serial.print("\tgyroY : ");

Serial.print(mpu6050.getGyroY());

Serial.print("\tgyroZ : ");

Serial.println(mpu6050.getGyroZ());

Serial.print("accAngleX : ");

Serial.print(mpu6050.getAccAngleX());

Serial.print("\taccAngleY : ");

Serial.println(mpu6050.getAccAngleY());

Serial.print("gyroAngleX : ");

Serial.print(mpu6050.getGyroAngleX());

Serial.print("\tgyroAngleY : ");

Serial.print(mpu6050.getGyroAngleY());

Serial.print("\tgyroAngleZ : ");

Serial.println(mpu6050.getGyroAngleZ());

Serial.print("angleX : ");

Serial.print(mpu6050.getAngleX());

Serial.print("\tangleY : ");

Serial.print(mpu6050.getAngleY());

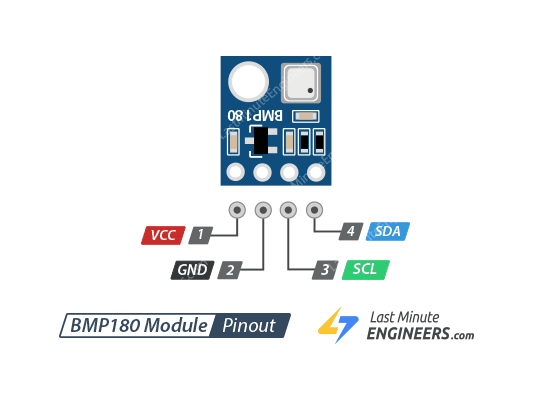
Serial.print("\tangleZ : ");

Serial.println(mpu6050.getAngleZ());

Serial.println("----------------------------------\n");

}

## **BMP180 Module**



VCC is the power supply for the module which can be anywhere between 3.3V to 5V.

GND should be connected to the ground of Arduino.

SCL is a serial clock pin for I2C interface.

SDA is a serial data pin for I2C interface.

Library: BMP180 – Adafruit BMP085 library

Code

#include <Wire.h>

#include <Adafruit\_BMP085.h>

#define seaLevelPressure\_hPa 1013.25

Adafruit\_BMP085 bmp;

void setup() {

Serial.begin(9600);

if (!bmp.begin()) {

Serial.println("Could not find a valid BMP085 sensor, check wiring!");

while (1) {}

}

}

void loop() {

Serial.print("Temperature = ");

Serial.print(bmp.readTemperature());

Serial.println(" \*C");

Serial.print("Pressure = ");

Serial.print(bmp.readPressure());

Serial.println(" Pa");

Serial.print("Altitude = ");

Serial.print(bmp.readAltitude());

Serial.println(" meters");

Serial.print("Pressure at sealevel (calculated) = ");

Serial.print(bmp.readSealevelPressure());

Serial.println(" Pa");

Serial.print("Real altitude = ");

Serial.print(bmp.readAltitude(seaLevelPressure\_hPa \* 100));

Serial.println(" meters");

Serial.println();

delay(500);

}

MQ-135 Gas Sensor Module

| **Feature** | **Description** |
| --- | --- |
| Operating Voltage | 2.5-5.0V |
| Detecting Concentration | 10ppm-300ppm for NH3 10ppm-1000ppm for Benzene 10ppm-300ppm for Alcohol |
| Load Resistance | Adjustable |
| Heater Resistance | 33Ω ± 5% |
| Heater Consumption | less than 800mW |
| Operating Temperature | -10 to 45°C |

| **MQ-135 Module** | **Arduino** |
| --- | --- |
| VCC | 5V |
| GND | GND |
| AO | A0 |
| DO | Pin 2 |

* Normal air returns approximately 100-150
* Alcohol returns approximately 700
* Lighter gas returns approximately 750

CODE

void setup()

{

Serial.begin(115200); // sets the serial port to 9600

pinMode(13, OUTPUT);

pinMode(2, INPUT);

}

void loop()

{

int sensorValue = analogRead(0); // read analog input pin 0

int digitalValue = digitalRead(2);

Serial.print("Air quality (Harmful gas) = ");

Serial.print(sensorValue);

Serial.println(" PPM");

}