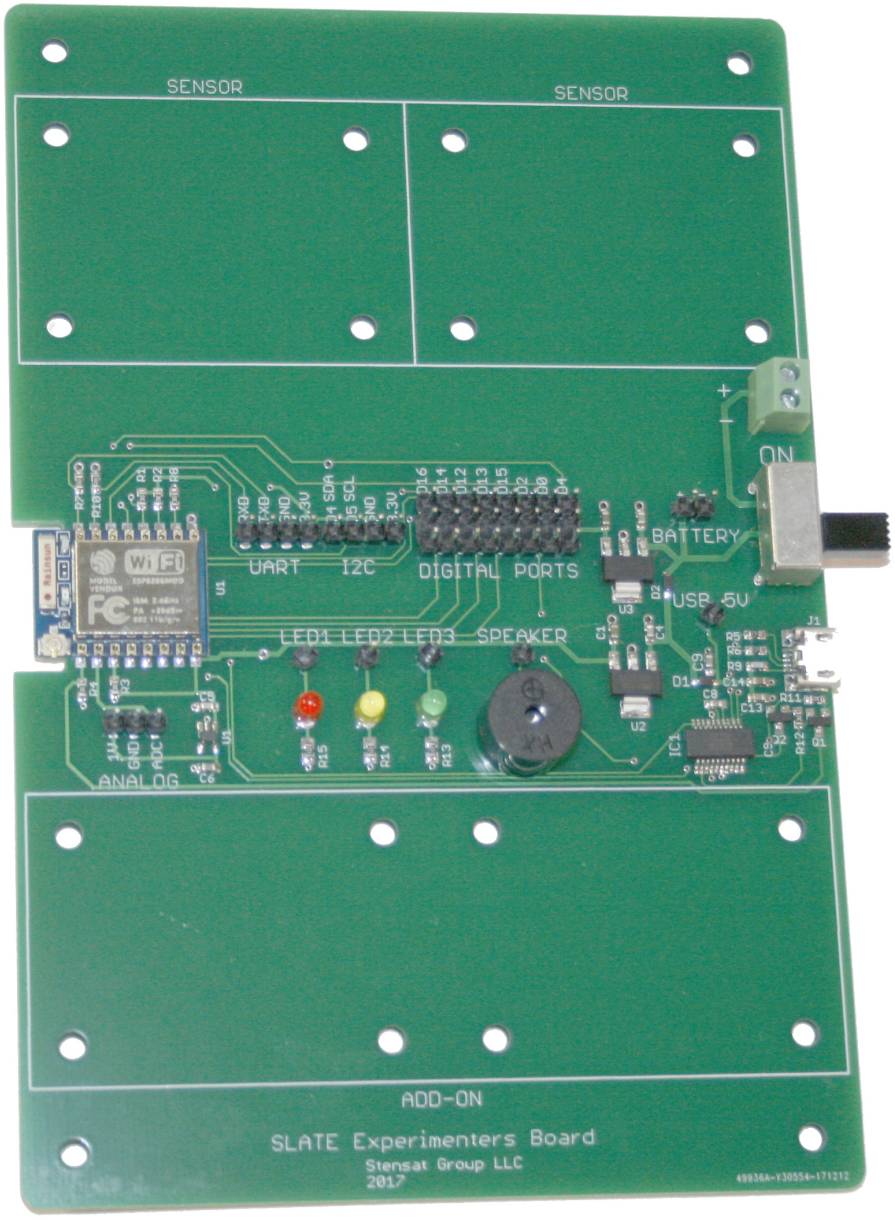
Sten-SLATE ESP Kit

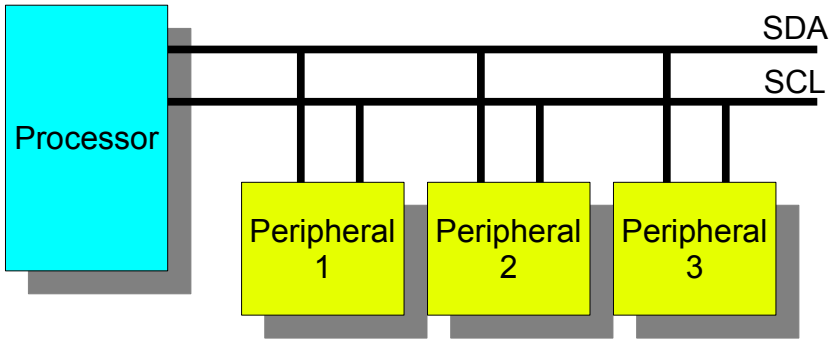
I2C Bus and IMU Sensor



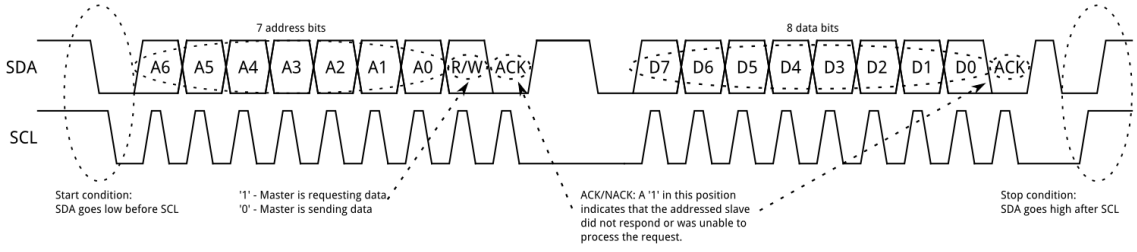
I2C Bus

I2C Bus

I2C stands for Inter-Integrated Circuit. It is a serial type interface requiring only two signals, a clock signal and a data signal. The I2C bus is typically used to interface with sensors and peripheral devices not needing to communicate at high speeds. The standard data rate is 100 Kilobits per second. Multiple devices can be connected to a single I2C bus. The processor is the controller and all the connected devices are peripherals. The processor is also called the master and the peripherals are slaves. Each slave has a unique address.



The clock signal is labeled SCL. This signal is used to control the flow of the data bits. The data signal is called SDA. This carries the data serially. The diagram below shows how a data transfer occurs. The data transfer protocol is for the master to first send out a device address. This is a 7 bit number followed by a bit indicating if the next byte is to be written to a slave or read from a slave. The SCL signal toggles for each bit sent.



I2C Sequence

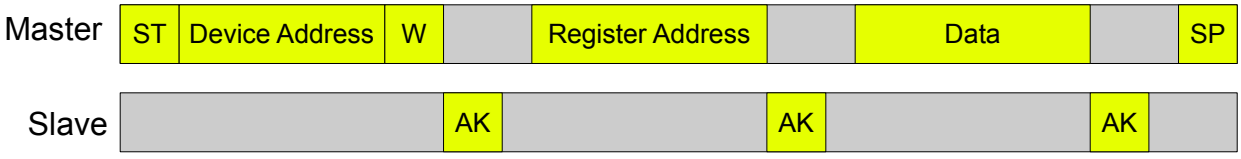
Every device on the I2C bus has a unique 7-bit address. The accelerometer address is 0x1C. The I2C operation for writing to a register is:
1. Send Start sequence by keeping SCL high and changing SDA from high to low (ST)

2. Send the device address

3. Send the register address

4. Send the stop sequence by changing SDA from low to high while SCL is high first. (SP)

All Bytes sent are acknowledge by the slave.



Every device on the I2C bus has a unique 7-bit address. The accelerometer address is 0x1C. The I2C operation for writing to a register is:

1. Send Start sequence by keeping SCL high and changing SDA from high to low (ST)

2. Send the device address

3. Send the register address

4. Send the stop sequence by changing SDA from low to high while SCL is high first. (SP)

All Bytes sent are acknowledge by the slave.

IMU

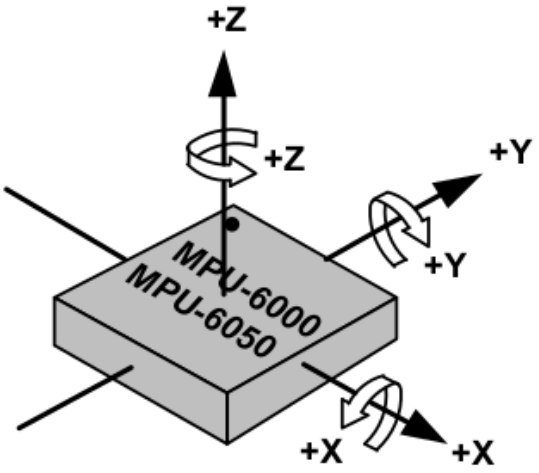
The MPU-6050 is a six degree of freedom Inertial Measurement Unit. It consists of a 3-axis accelerometer and a 3-axis rate gyroscope. The accelerometer has a settable sensitivity range of 2Gs to 16 Gs. The rate gyroscope has a settable sensitivity range of 250 degrees per second to 5000 degrees per second.

The sensor allows you to measure acceleration and rotation in three dimensions. With the combination of the the accelerometers and rate gyros, orientation can be measured in the three axis. In the X and Y axis, the absolute orientation relative to earth's gravity can be measured. The sensor will measure a range of -180 to +180 degrees.

In the Z axis, the accelerometer cannot be used so there is no absolute reference for the orientation around the Z axis. The zero angle orientation around the Z axis is established when the sensor is calibrated. As the sensor is rotated, the angle measurement accumulates. The measurement will not reset when exceeding -360 or +360 degrees. The number of rotations can be calculated by dividing the measured value by 360.

All gyros have drift which needs to be calibrated out. The programs later include a calibration mode. When the program starts, the sensor has to not move at all. The calibration will take about three seconds and will measure the gyro drifts so they can be subtracted from the measurements.

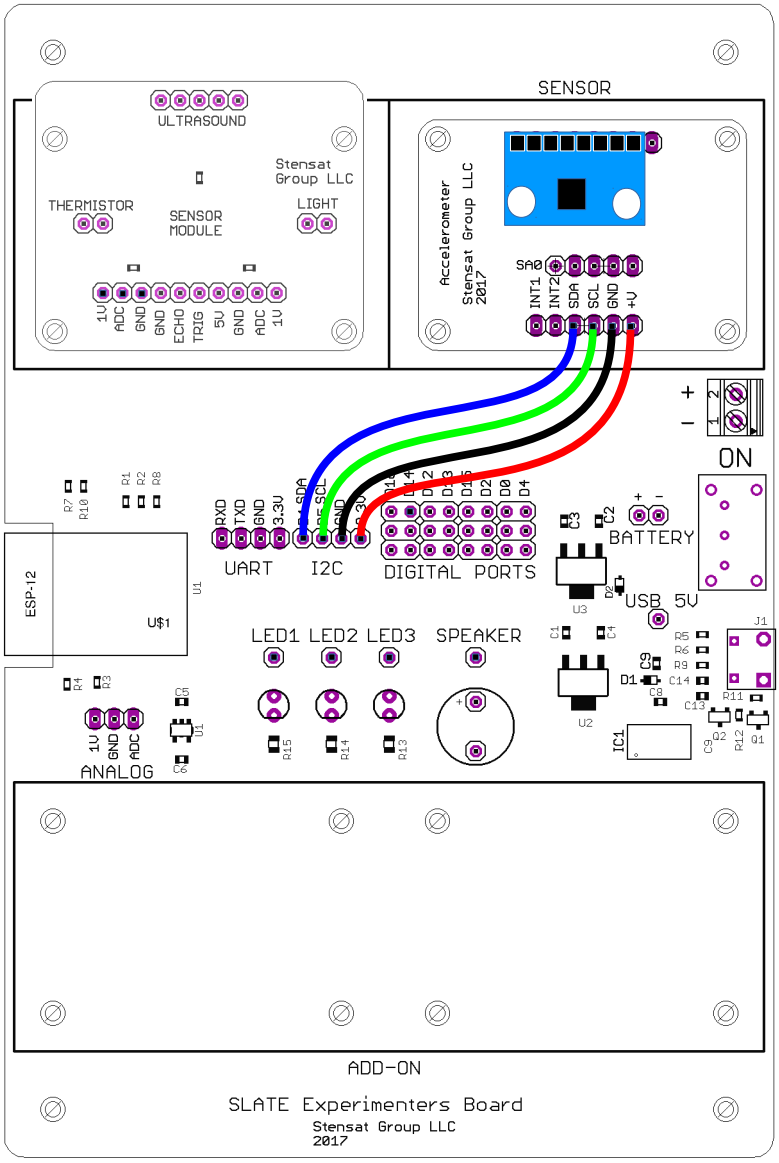
In this lesson, the rate gyro will be used to determine orientation. The library includes functions for calculating the angle of the sensor after it is calibrated. A processing program will be used to graphically demonstrate the orientation of the sensor.



Sensor Axis Diagram

IMU

Install the sensor board into the second sensor location as shown. Secure it with nuts. Connect the jumpers as shown. Connect from the pins above the I2C label on the experimenters board.



Connect 3.3V to V+. Connect both GND signals. Connect D5 to SCL. Connect D4 to SDA.

Download the library from www.stenat.org www.stensat.org. This library was modified to work properly on the ESP8266 SLATE board. In the Arduino IDE, select the **Sketch** menu and then select **Include Library**. Select **Add .ZIP file**. Locate the file and select it. It will be added to your library. In the **File** menu, select **Examples** then locate **MPU6050_tockn**. Select **GetAllData**. The program will open in a window.

Before compiling, make the following changes as shown in the next page:

Line 11, **Wire.begin(); => Wire.begin(4,5);**

Line 12, **mpu6050.begin(); ==> mpu6050.begin(ACCEL_2G,GYRO_500);**

Change line 10 baud rate from 9600 to 115200.

Compile and upload the program.

When done uploading, open the Serial monitor and make sure the baud rate is set to 115200. When the program starts, it will spend about 3 seconds calibrating. Make sure the sensor is not moving during this time. It will calibrate right after the code finishes uploading.

GetAllData Program

```
#include <MPU6050_tockn.h>
#include <Wire.h>

MPU6050 mpu6050(Wire);

long timer = 0;

void setup() {
  Serial.begin(115200);
  Wire.begin(4,5);
  mpu6050.begin(ACCEL_2G,GYRO_500);
  mpu6050.calcGyroOffsets(true);
}

void loop() {
  mpu6050.update();

  if(millis() - timer > 1000){

    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();

  }

}
```

In the example program, the sensor library is included at line 2. Line 3 loads the I2C library. Line 5 creates a sensor object. The argument is Wire which tells the library to use the I2C interface. This is done to allow multiple I2C buses to be used. Only one is used here.

The timer variable in line 7 is used to track the time and have the display updated sensor data once a second. Lines 9-14 is the setup function. The serial interface is configued then the I2C interface. Next the sensor is configured with the accelerometer set to 2G range and the gyro set to 500 degrees per second rotation rate range. Line 13 calls a library function to calibrate the gyroscope. The gyroscope has what is called a DC offset or constant offset. This is an error that all sensors have and can be measured with the sensor not moving. The library subtracts the offset from all measurements.

Lines 16 – 46 is the loop function. Line 19 determines if a second has passed. If so, the reset of the code is executed. Line 17 is the function that collects the sensor data. The results are kept in the library variables. Lines 22-40 display the sensor results Notice that the values displayed are function calls. **mpu6050.getTemp()** will return the temperatuer in Celcius. **mpu6050.getAccx()** will return the X-axis accelerometer value in Gs and so on. Notice the values are in floating point and processed from the raw values. Lines 31 and 32 return the sensor angle in the X and Y axis based on the accelerometer.

mpu6050.getAccAngleX() returns an angle in degrees referenced to the Z and X axis.

mpu6050.getAccAngleY() returns the angle in degrees referenced to the Z and Y axis.

mpu6050.getGyroAngleX() returns the angle calculated by the accumulation of the rate gyro around the X axis.

mpu6050.getGyroAngleY() returns the angle calculated by the accumulation of the rate gyro around the Y axis.

mpu6050.getGyroAngleZ() returns the angle calculated by the accumulation of the rate gyro around the Z axis.

mpu6050.getAngleX() provides the angle around the X axis based on the combination of the accelerometer and gyro.

mpu6050.getAngleY() provides the angle around the Y axis based on the combination of the accelerometer and gyro.

mpu6050.getangleZ() provides the angle around the Z axis based on the combination of the accelerometer and gyro.

These three functions provide the best orientation value of the sensor and can be used to indicate the orientation of any device it is connected.

Simpler IMU Program

This program is a simpler version of the example program where only the X,Y,Z angles are sent over the USB port.

Enter this program in the Arduino IDE and upload to the SLATE. Name the program **simpleimu.**

Instead of opening the Serial Monitor, select the menu **Tools** and select **Serial Plotter**. A window will open and plot three lines as data is streaming.

Rotate the board on the table and see how the Z-axis data changes. Rotate more than 360 degrees. Notice the plot goes out of range. Rotate in the opposite direction. The Z-axis plot should return back into the plot area.

Do the same for the other two axis. Notice they only go to +/- 180 degrees. The accelerometer is being used with the gyro to maintain orientation information. The accelerometers are using gravity as a reference to down. For the Z-axis, there is no reference. A magnetometer could be used as a reference for the Z-axis using the earth's magnetic field.

Save this program. It will be used later.

```
#include <MPU6050_tockn.h>
#include <Wire.h>

MPU6050 mpu6050(Wire);
long timer = 0;


void setup() {
  Serial.begin(115200);
  Wire.begin(4,5);
  mpu6050.begin(ACCEL_2G,GYRO_500);
  mpu6050.calcGyroOffsets(true);
}

void loop() {
  mpu6050.update();
  Serial.print(mpu6050.getAngleX());
  Serial.print(",");
  Serial.print(mpu6050.getAngleY());
  Serial.print(",");
  Serial.println(mpu6050.getAngleZ());
  delay(10);
}
```