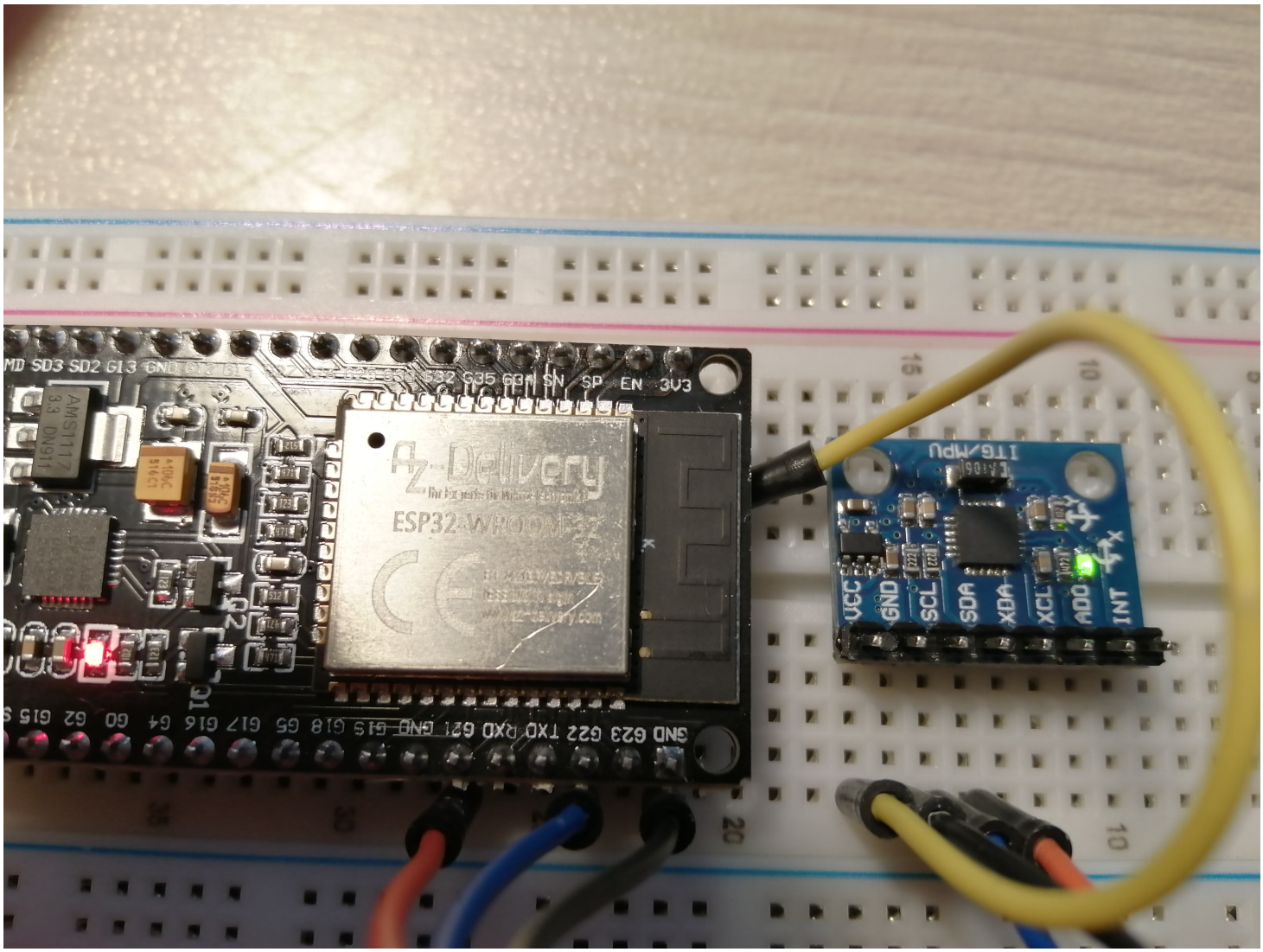
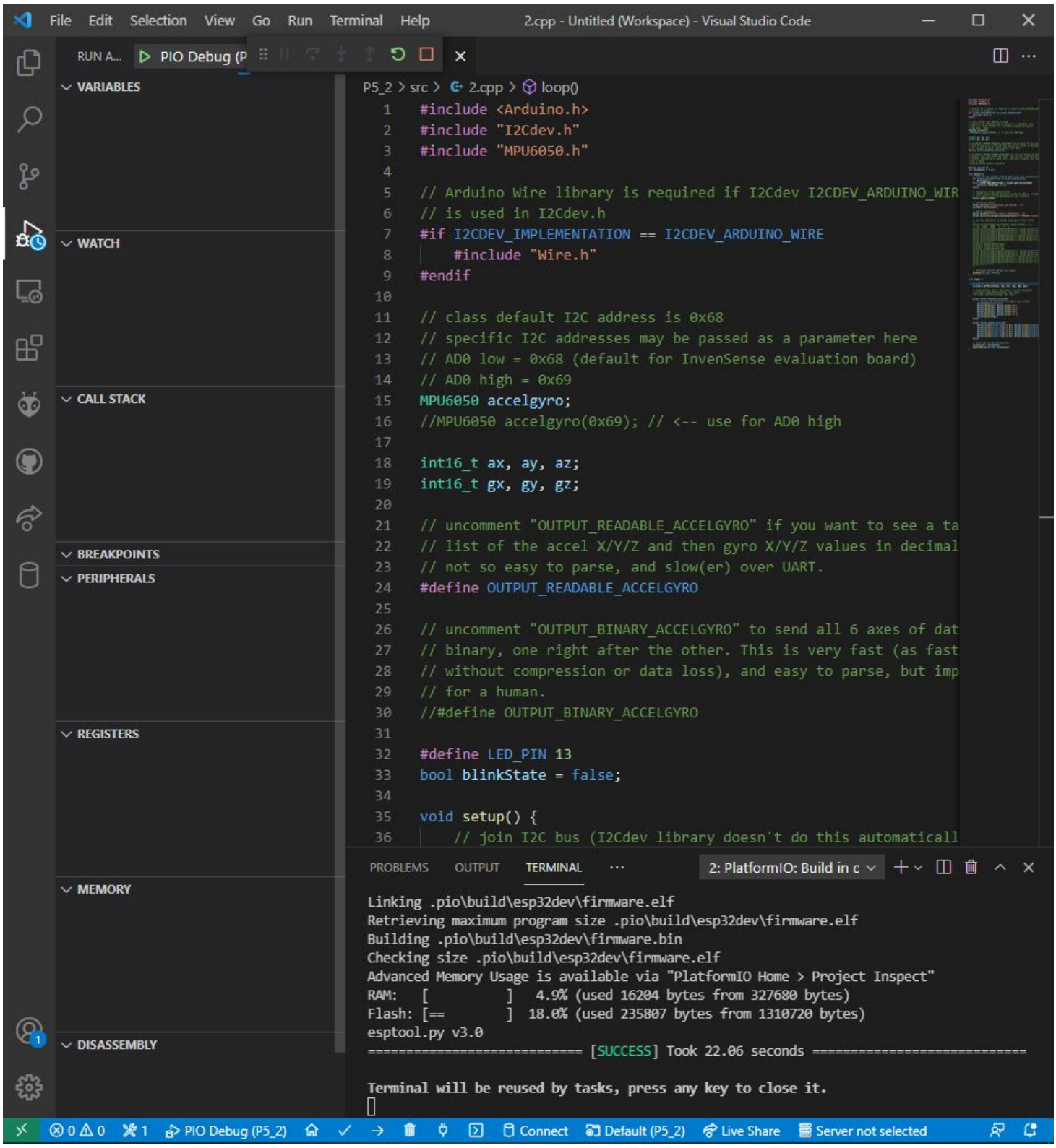
PRACTICA 5_2 : Buses de comunicación I (introducción y I2c)

1.Fotos del montaje



2.Salidas de depuración (print...)



3.Código generado

```
#include <Arduino.h>
#include "I2Cdev.h"
#include "MPU6050.h"

// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    #include "Wire.h"
#endif

// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for InvenSense evaluation board)
// AD0 high = 0x69
MPU6050 accelgyro;
//MPU6050 accelgyro(0x69); // <-- use for AD0 high

int16_t ax, ay, az;
int16_t gx, gy, gz;

// uncomment "OUTPUT_READABLE_ACCELGYRO" if you want to see a tab-separated
// list of the accel X/Y/Z and then gyro X/Y/Z values in decimal. Easy to read,
// not so easy to parse, and slow(er) over UART.
#define OUTPUT_READABLE_ACCELGYRO

// uncomment "OUTPUT_BINARY_ACCELGYRO" to send all 6 axes of data as 16-bit
// binary, one right after the other. This is very fast (as fast as possible
// without compression or data loss), and easy to parse, but impossible to read
// for a human.
//#define OUTPUT_BINARY_ACCELGYRO

#define LED_PIN 13
bool blinkState = false;

void setup() {
    // join I2C bus (I2Cdev library doesn't do this automatically)
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif

    // initialize serial communication
    // (38400 chosen because it works as well at 8MHz as it does at 16MHz, but
    // it's really up to you depending on your project)
    Serial.begin(115200);

    // initialize device
    Serial.println("Initializing I2C devices...");
    accelgyro.initialize();
```

```
    // verify connection
    Serial.println("Testing device connections...");
    Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful" : "MPU6050 connection failed");

    // use the code below to change accel/gyro offset values
    /*
    Serial.println("Updating internal sensor offsets...");
    // -76      -2359   1688    0       0       0
    Serial.print(accelgyro.getXAccelOffset()); Serial.print("\t"); // -76
    Serial.print(accelgyro.getYAccelOffset()); Serial.print("\t"); // -2359
    Serial.print(accelgyro.getZAccelOffset()); Serial.print("\t"); // 1688
    Serial.print(accelgyro.getXGyroOffset()); Serial.print("\t"); // 0
    Serial.print(accelgyro.getYGyroOffset()); Serial.print("\t"); // 0
    Serial.print(accelgyro.getZGyroOffset()); Serial.print("\t"); // 0
    Serial.print("\n");
    accelgyro.setXGyroOffset(220);
    accelgyro.setYGyroOffset(76);
    accelgyro.setZGyroOffset(-85);
    Serial.print(accelgyro.getXAccelOffset()); Serial.print("\t"); // -76
    Serial.print(accelgyro.getYAccelOffset()); Serial.print("\t"); // -2359
    Serial.print(accelgyro.getZAccelOffset()); Serial.print("\t"); // 1688
    Serial.print(accelgyro.getXGyroOffset()); Serial.print("\t"); // 0
    Serial.print(accelgyro.getYGyroOffset()); Serial.print("\t"); // 0
    Serial.print(accelgyro.getZGyroOffset()); Serial.print("\t"); // 0
    Serial.print("\n");
    */

    // configure Arduino LED pin for output
    pinMode(LED_PIN, OUTPUT);
}

void loop() {
    // read raw accel/gyro measurements from device
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    // these methods (and a few others) are also available
    //accelgyro.getAcceleration(&ax, &ay, &az);
    //accelgyro.getRotation(&gx, &gy, &gz);

    #ifdef OUTPUT_READABLE_ACCELGYRO
        // display tab-separated accel/gyro x/y/z values
        Serial.print("a/g:\t");
        Serial.print(ax); Serial.print("\t");
        Serial.print(ay); Serial.print("\t");
        Serial.print(az); Serial.print("\t");
        Serial.print(gx); Serial.print("\t");
        Serial.print(gy); Serial.print("\t");
        Serial.println(gz);
    #endif

    #ifdef OUTPUT_BINARY_ACCELGYRO
```

```
        Serial.write((uint8_t)(ax >> 8)); Serial.write((uint8_t)(ax & 0xFF));
        Serial.write((uint8_t)(ay >> 8)); Serial.write((uint8_t)(ay & 0xFF));
        Serial.write((uint8_t)(az >> 8)); Serial.write((uint8_t)(az & 0xFF));
        Serial.write((uint8_t)(gx >> 8)); Serial.write((uint8_t)(gx & 0xFF));
        Serial.write((uint8_t)(gy >> 8)); Serial.write((uint8_t)(gy & 0xFF));
        Serial.write((uint8_t)(gz >> 8)); Serial.write((uint8_t)(gz & 0xFF));
    #endif

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

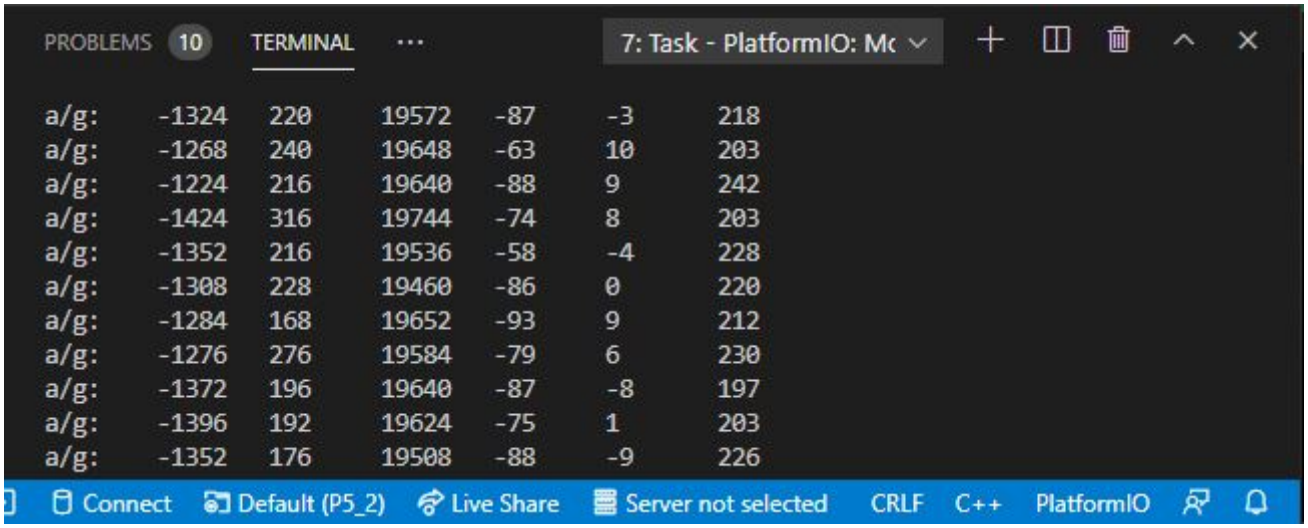
4.Explicación del código

El código de este ejercicio se basa en la función del I2c MPU6050, que es medir la aceleración y la rotacióon del propio sensor. Para ello, se utilizan 3 variables que simulan las componentes x, y, z de un espacio tridimensional. Para captar el movimiento del sensor, se utiliza la función "getMotion6()".

De modo que, a medida que variamos la dinámica del movimiento del sensor, los datos de cada componente de la aceleración y rotación van cambiando en el terminal de salida en función del tiempo.

5.Salida del terminal

5.1.Protoboard horizontal:



5.2.Protoboard vertical:

PROBLEMS 10 TERMINAL ··· 7: Task - PlatformIO: Mc

```
a/g:    -548    -16500  3136    735     540     606
a/g:    -588    -16400  3060    939     520     464
a/g:    -496    -16552  3092    1096    522     371
a/g:    -444    -16420  3148    1200    527     270
a/g:    -388    -16412  3216    1313    487     181
a/g:    -424    -16404  3352    1376    435     107
a/g:    -364    -16316  3532    1413    373     25
a/g:    -268    -16284  3356    1467    326     -4
a/g:    -272    -16340  3516    1440    278     -61
a/g:    -332    -16320  3564    1397    202     -111
a/g:    -344    -16428  3576    1311    202     -142
```

Connect Default (P5_2) Live Share Server not selected CRLF C++ PlatformIO