TÉCNICO LISBOA

DEEC
DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Project Rocket Payload

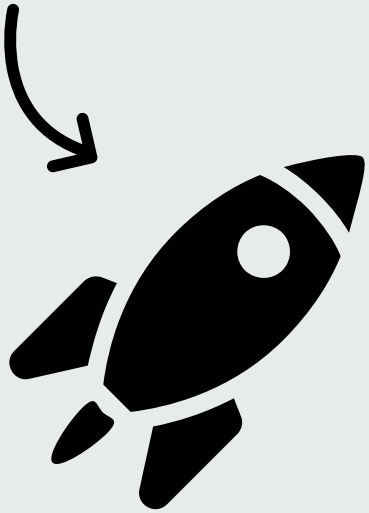
SEP

Antónia Espírito Santo
Mariana Lopes Teixeira
Pedro Carvalho Rodrigues

TÉCNICO LISBOA

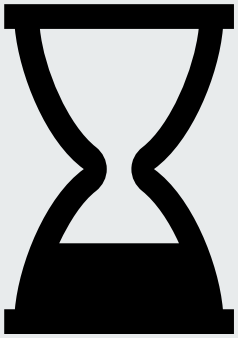
Motivation

- Why did we propose this project?
- Objectives

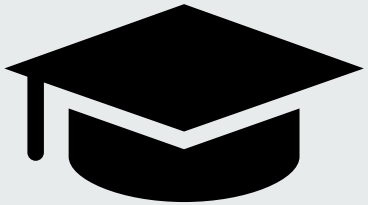


TÉCNICO LISBOA

Problems that changed everything…



Short time

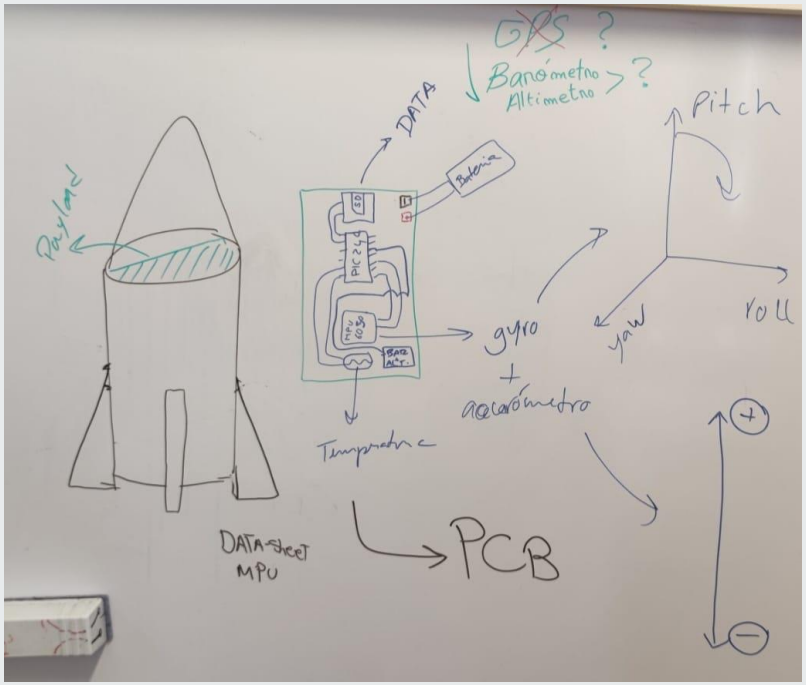


Very complex

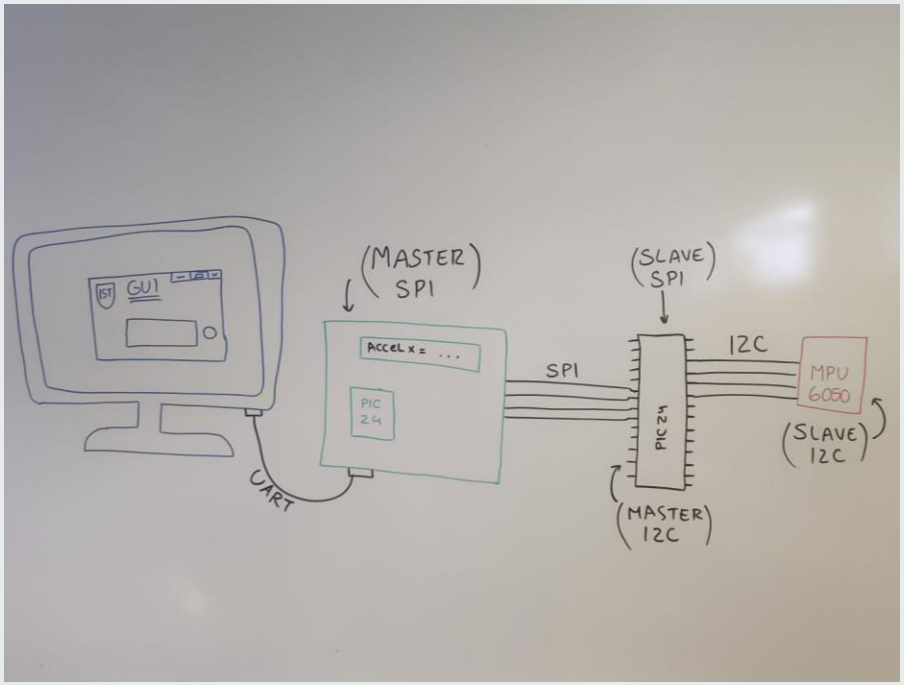
Other classes and projects

Overall project

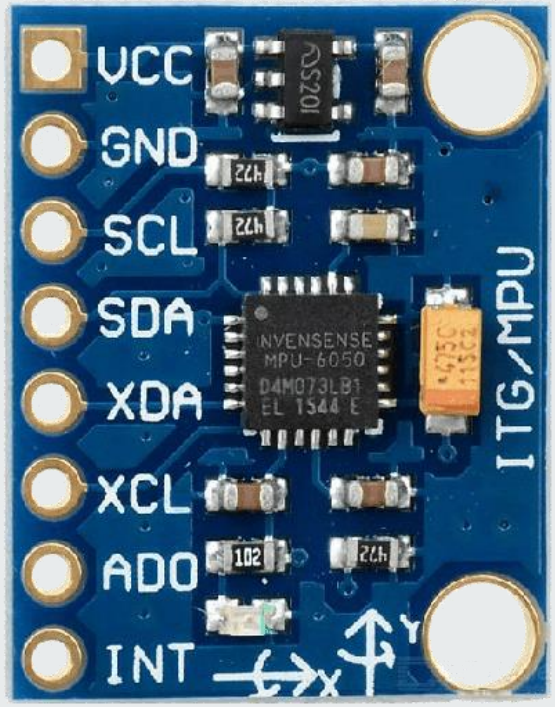
- Before:



- After:

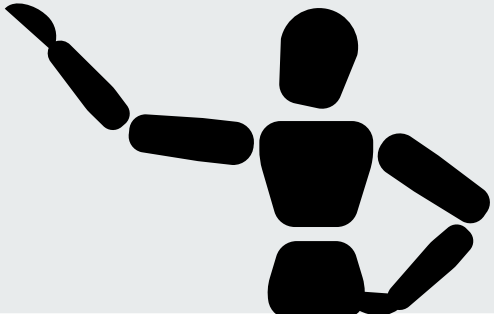


Sensor



MPU6050

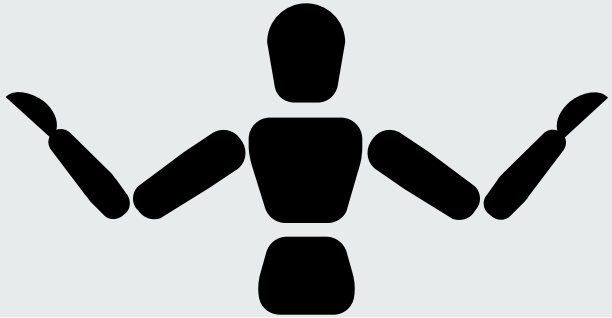
- Accelerometer (X,Y and Z axis)
- Gyroscope (X,Y and Z axis)
- Internal Temperature



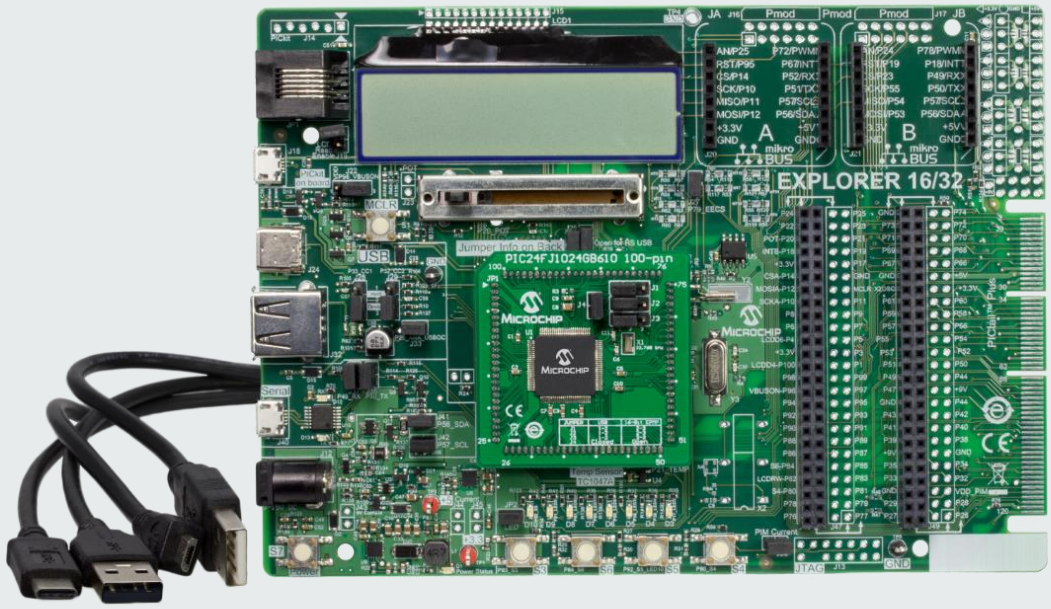
Sensor Node

PIC24FJ256GA702

- 16-bit microcontroller
- 256 KB of Flash program memory
- I2C, SPI, UART, …



Serial/SPI Bridge

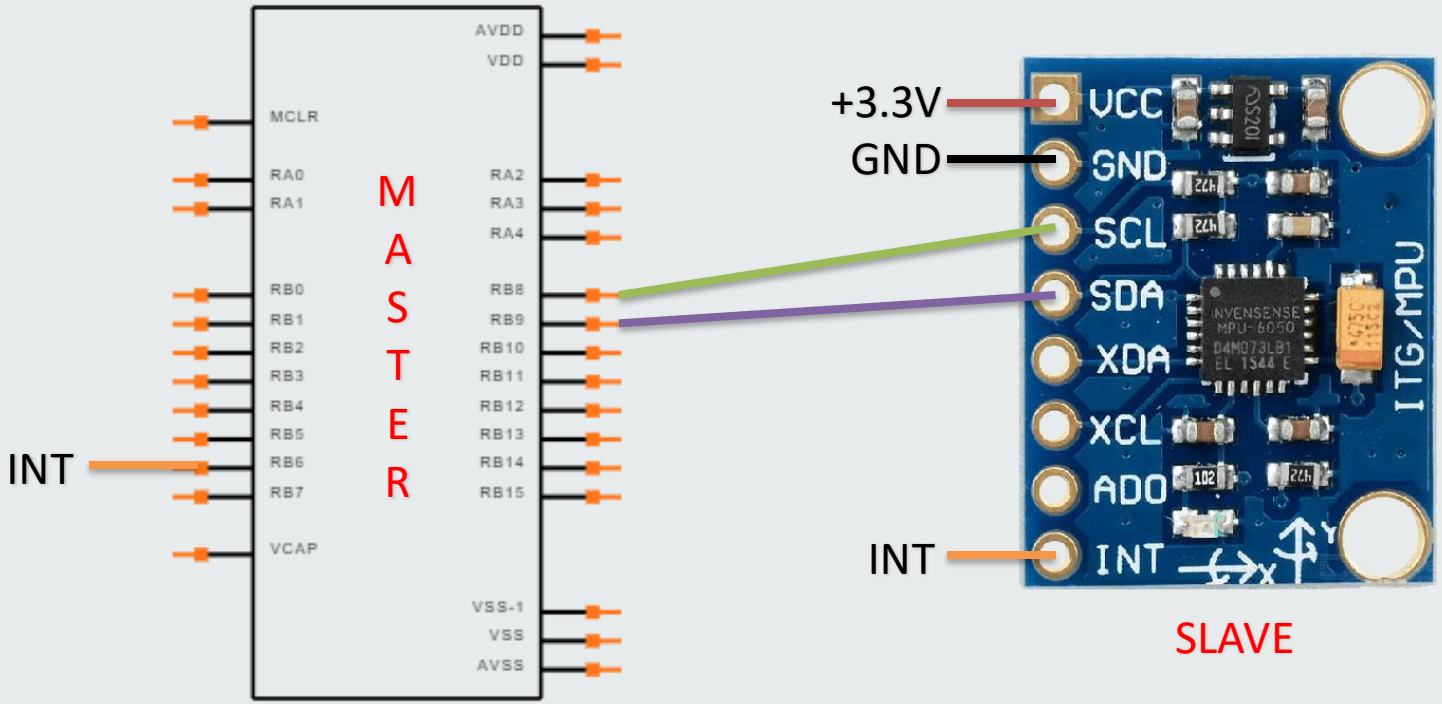


PIC24FJ1024GB610

- 16-bit microcontroller
- 1024 KB of Flash program memory

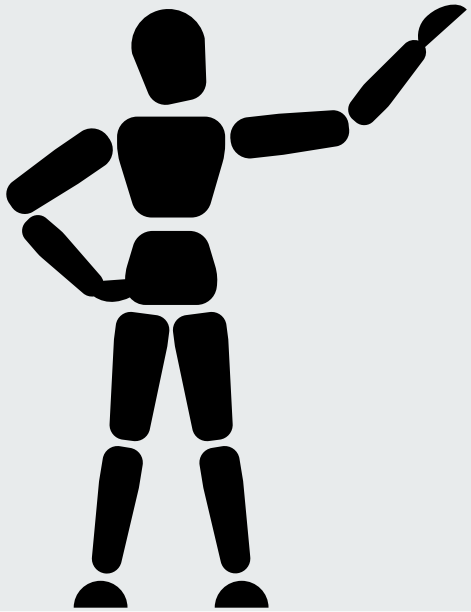
I2C

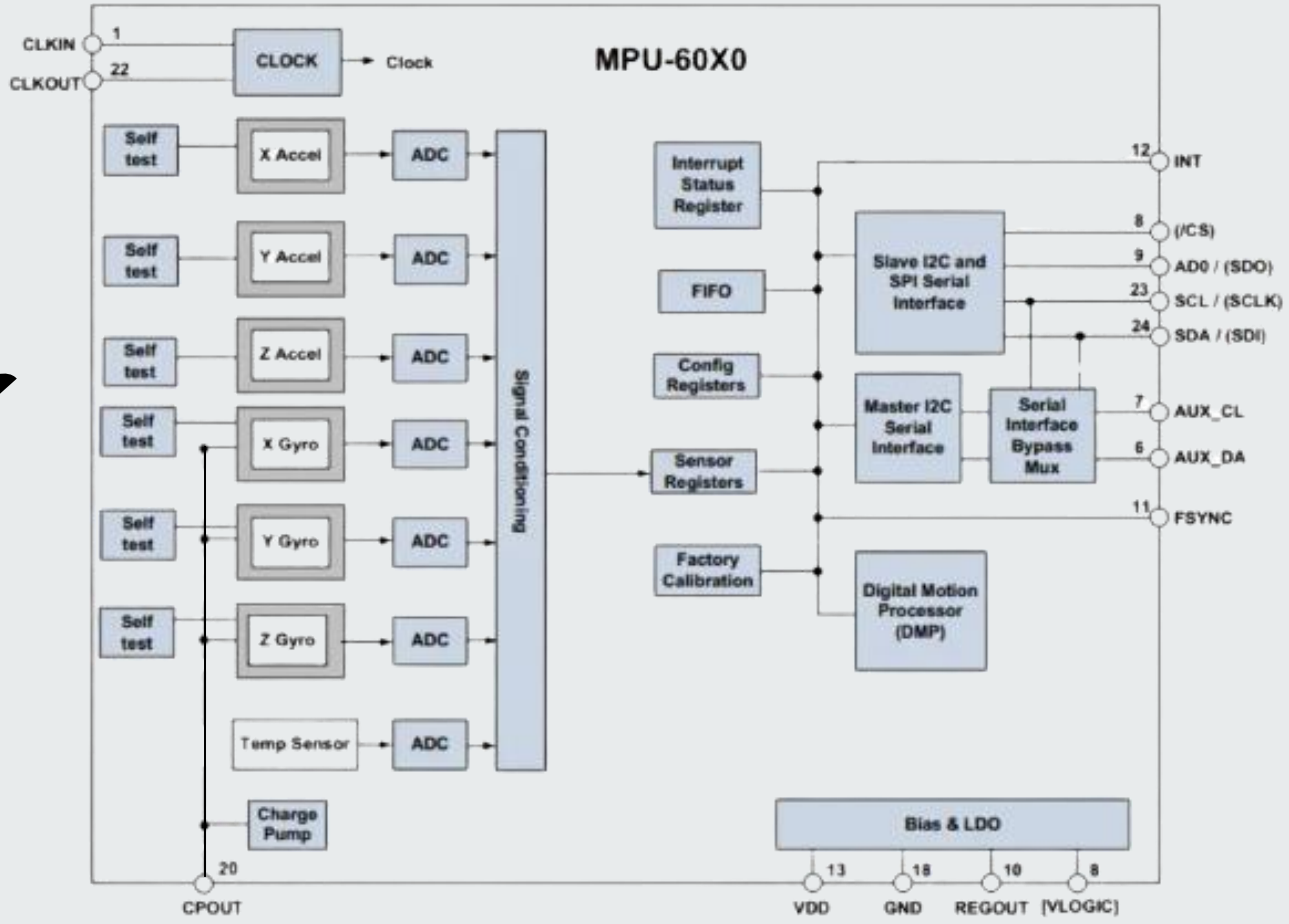
- Used between PIC24FJ256GA702 and MPU6050



TÉCNICO LISBOA

I2C





TÉCNICO LISBOA

I2C

4.32 Register 117 – Who Am I
WHO_AM_I

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
75	117	-	WHO_AM_I[6:1]						-

I2C

Burst Read Sequence

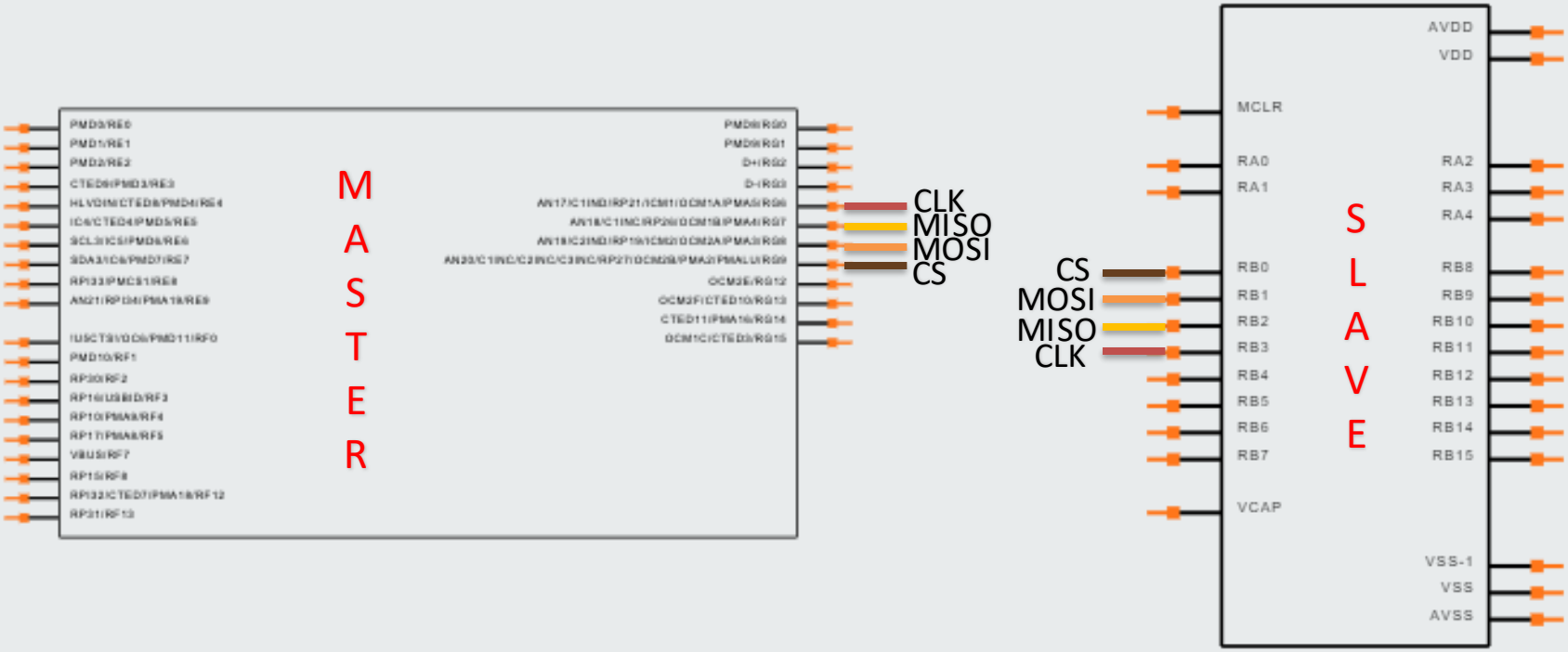
Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

```
#define SMPLRT_DIV 0x19
#define CONFIG 0x1A
#define GYRO_CONFIG  0x1B
#define ACCEL_CONFIG 0x1C
#define INT_PIN_CFG 0x37
#define INT_ENABLE 0x38
#define DMP_INT_STATUS 0x39
#define INT_STATUS 0x3A
#define ACCEL_XOUT 0x3B
#define ACCEL_YOUT 0x3D
#define ACCEL_ZOUT 0x3F
#define GYRO_XOUT 0x43
#define GYRO_YOUT 0x45
#define GYRO_ZOUT 0x47
#define MPU6050_ADD 0x68
#define PWR_MGMT_1 0x6B
#define REGISTER_ADD 0x75
#define MPU6050_ADD_W 0xD0
#define MPU6050_ADD_R 0xD1
```

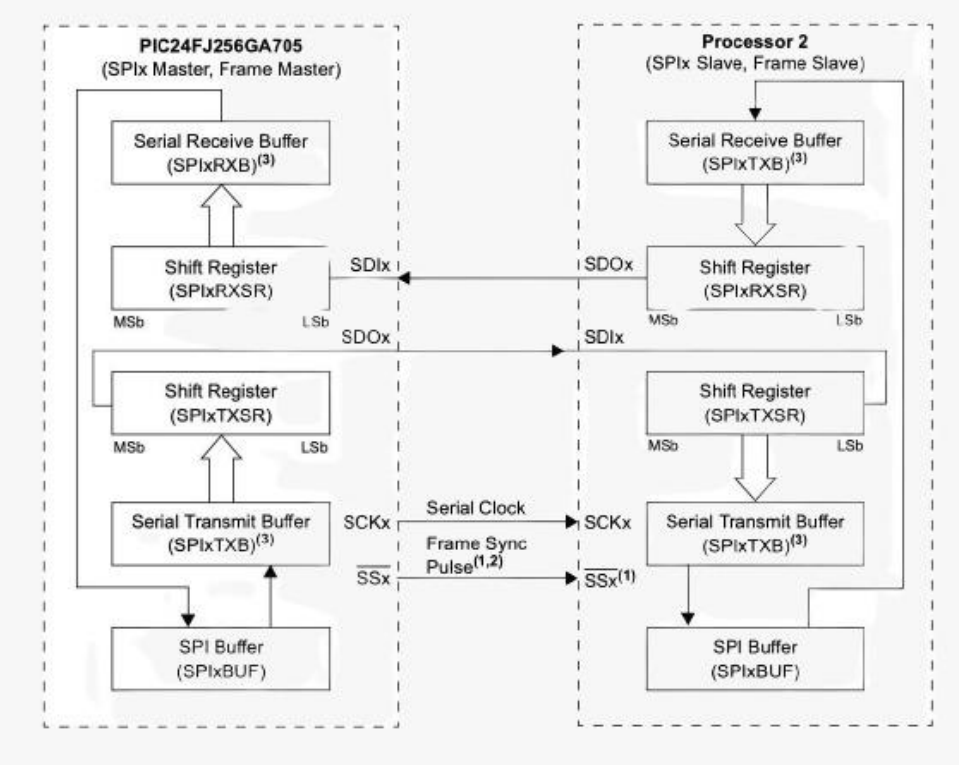
```
void MPU6050()
{
    // Start I2C communication
    startConditionI2C();

    // Send the device address with write operation
    writeByteI2C(MPU6050_ADD_W);

    // Send the register address to read from
    writeByteI2C(ACCEL_XOUT);

    startConditionI2C();

    writeByteI2C(MPU6050_ADD_R);

    Ax = (((uint16_t) readByteI2C(0))<<8) | (uint16_t) readByteI2C(0);
    Ay = (((uint16_t) readByteI2C(0))<<8) | (uint16_t) readByteI2C(0);
    Az = (((uint16_t) readByteI2C(0))<<8) | (uint16_t) readByteI2C(0);
    T  = (((uint16_t) readByteI2C(0))<<8) | (uint16_t) readByteI2C(0);
    Gx = (((uint16_t) readByteI2C(0))<<8) | (uint16_t) readByteI2C(0);
    Gy = (((uint16_t) readByteI2C(0))<<8) | (uint16_t) readByteI2C(0);
    Gz = (((uint16_t) readByteI2C(0))<<8) | (uint16_t) readByteI2C(1);

    stopConditionI2C();

    // Convert The Readings
    AX = (float)Ax/16384.0;
    AY = (float)Ay/16384.0;
    AZ = (float)Az/16384.0;
    GX = (float)Gx/131.0;
    GY = (float)Gy/131.0;
    GZ = (float)Gz/131.0;
    t = ((float)T/340.00)+36.53;
}
```

SPI

- Used between PIC24FJ256GA702 and PIC24FJ1024GB610



SPI


PIC24FJ256GA705
(SPIx Master, Frame Master)
Processor 2
(SPIx Slave, Frame Slave)
Serial Receive Buffer (SPIxRXB)(3)
Shift Register (SPIxRXSR)
MSb
LSb
SDIx
SDOx
Shift Register (SPIxTXSR)
Serial Transmit Buffer (SPIxTXB)(3)
SCKx
Serial Clock
Frame Sync Pulse(1,2)
SSx
SSx(1)
SPI Buffer (SPIxBUF)

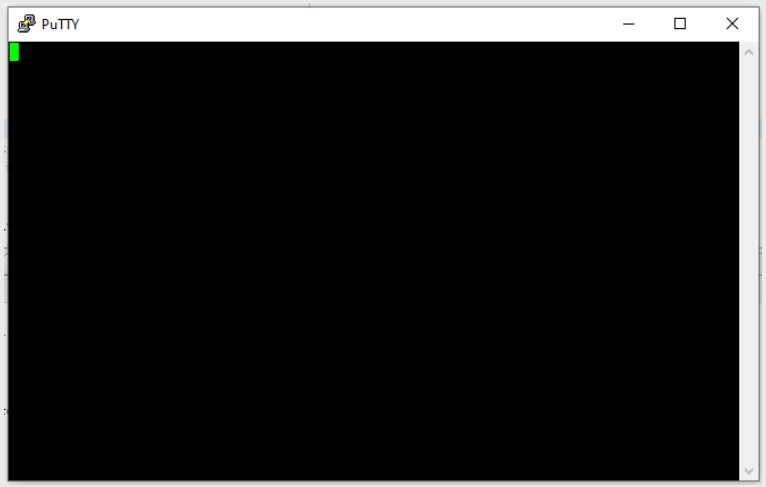
TÉCNICO LISBOA

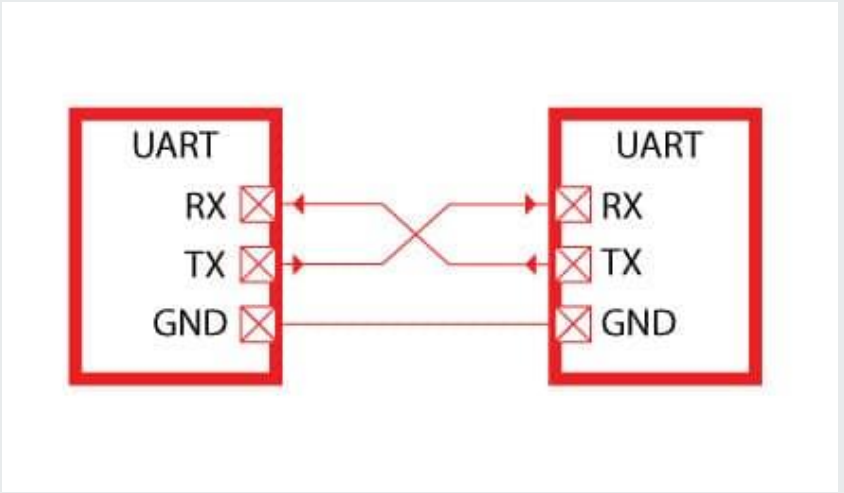
SPI

```
LATGbits.LATG9 = 0;

__delay_ms(10);

writeByteSPI(AX);
writeByteSPI(DUMMY);
atain[0] = writeByteSPI(DUMMY);
datain[1] = writeByteSPI(DUMMY);

datain_0 = (uint32_t*)&datain[0];
datain_1 = (uint32_t*)&datain[1];
value = (datain_0 << 16) | datain_1;

data = *((float*)&datain_0);

__delay_ms(10);

LATGbits.LATG9 = 1;
```

```
void sendCommand(){

    uint16_t command;

    uint16_t buffer;
    uint16_t dataOut[2];
    uint16_t* parameterHex;
    float parameter; // value after MPU has been processed

    SPI1BUFL = DUMMY;
    while(!SPI1STATLbits.SPIRBF); // wait for transfer to complete
    command = SPI1BUFL; // command, what we're going to receive
    while(SPI1STATLbits.SPITBF); // wait for transfer to complete


    SPI1BUFL = DUMMY;
    while(!SPI1STATLbits.SPIRBF); // wait for transfer to complete
    buffer = SPI1BUFL; // DUMMY
    while(SPI1STATLbits.SPITBF); // wait for transfer to complete


        if (command == AX){parameter = ax;}
    else if (command == AY){parameter = ay;}
    else if (command == AZ){parameter = az;}
    else if (command == GX){parameter = gx;}
    else if (command == GY){parameter = gy;}
    else if (command == GZ){parameter = gz;}
    else if (command == TI){parameter = ti;}
    else {return;} // command will never be DUMMY


    converts from float to hex
    parameterHex = (uint32_t*)&parameter;
    dataOut[0] = (uint16_t)(*parameterHex);
    dataOut[1] = (uint16_t)(*parameterHex >> 16);


    // sends the data in two parts
    dataOut[1]=10;
    SPI1BUFL = dataOut[1];
    while(!SPI1STATLbits.SPIRBF); // wait for transfer to complete
    buffer = SPI1BUFL; // DUMMY
    while(SPI1STATLbits.SPITBF); // wait for transfer to complete

    SPI2BUFL = dataOut[0];
    while(!SPI2STATLbits.SPIRBF); // wait for transfer to complete
    buffer = SPI2BUFL; // DUMMY
    while(SPI2STATLbits.SPITBF); // wait for transfer to complete
}
```

UART



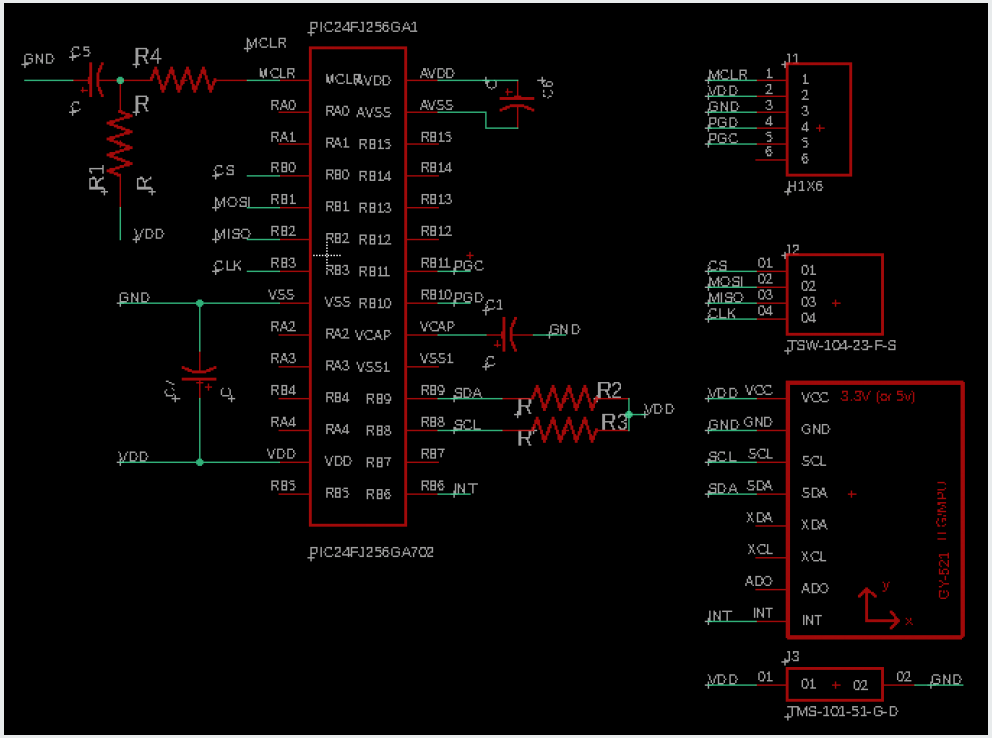


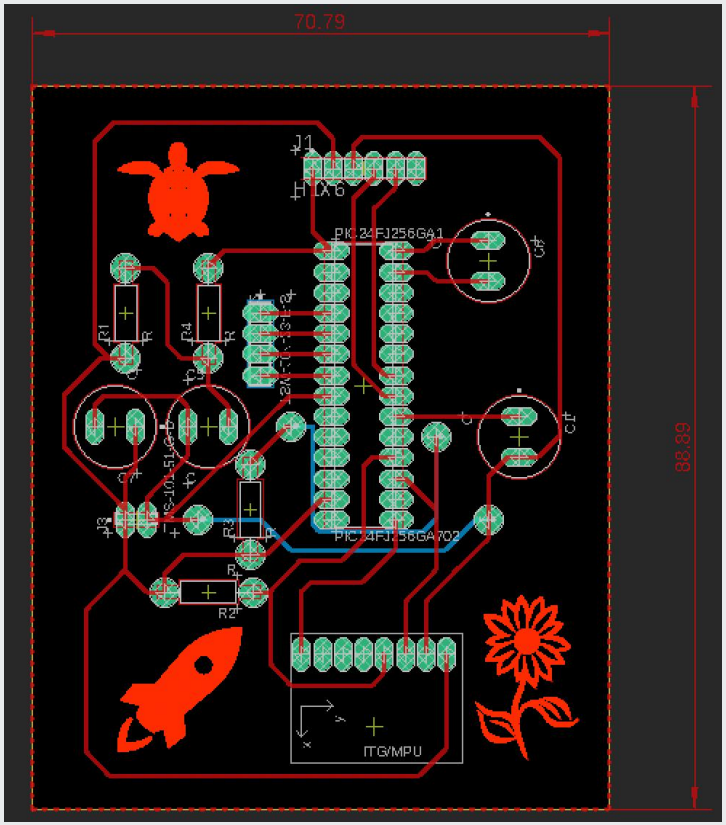
The problem with Pick-it 3

- Not enough current…

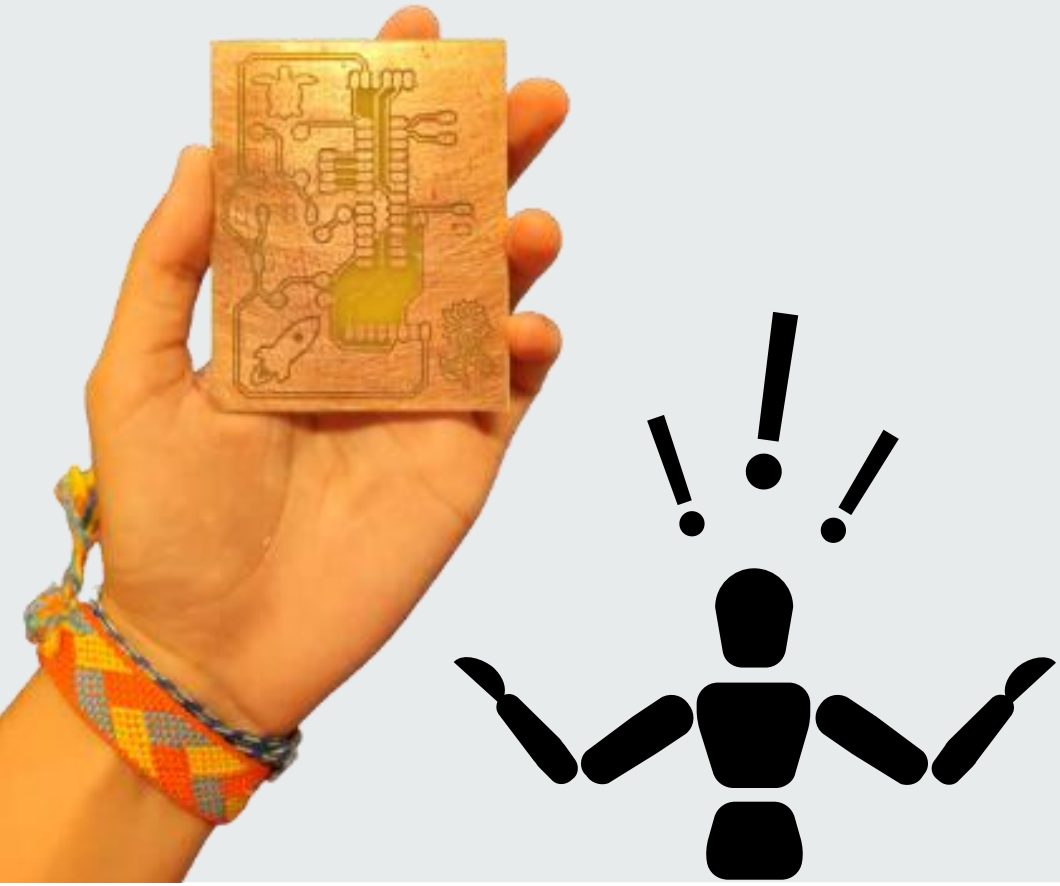
TÉCNICO LISBOA

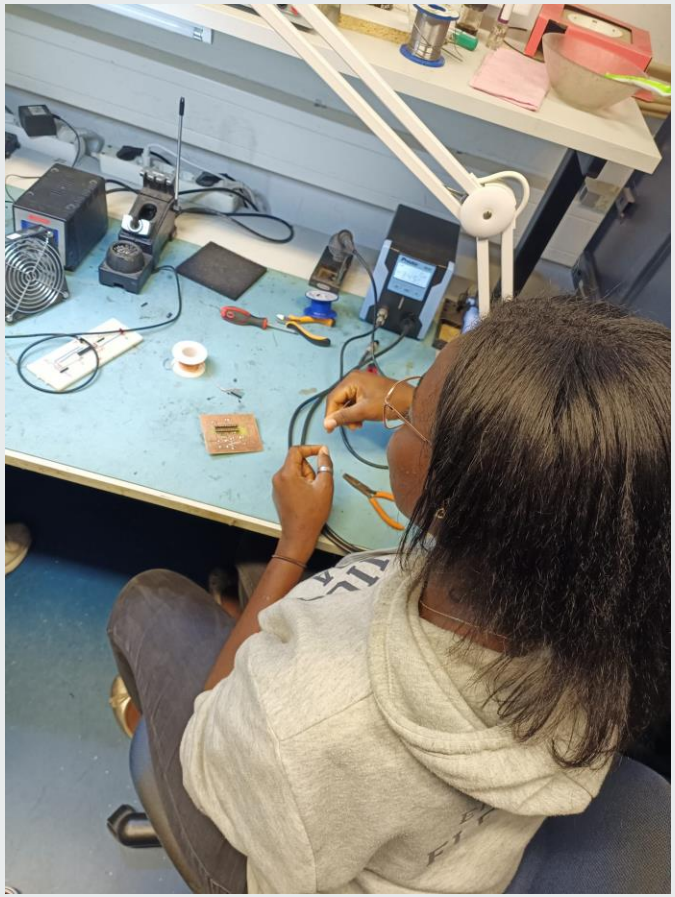
Schematic and Layout





PCB Production

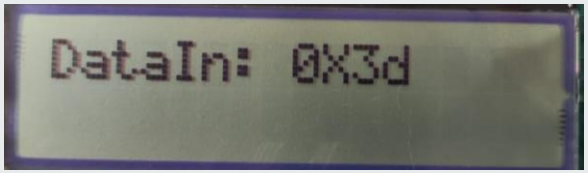


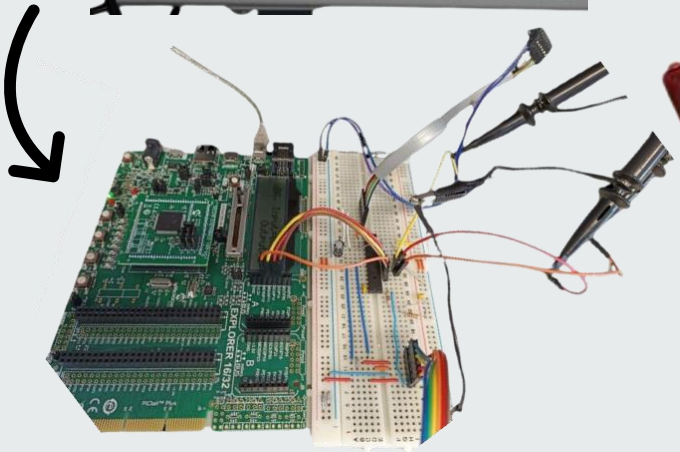


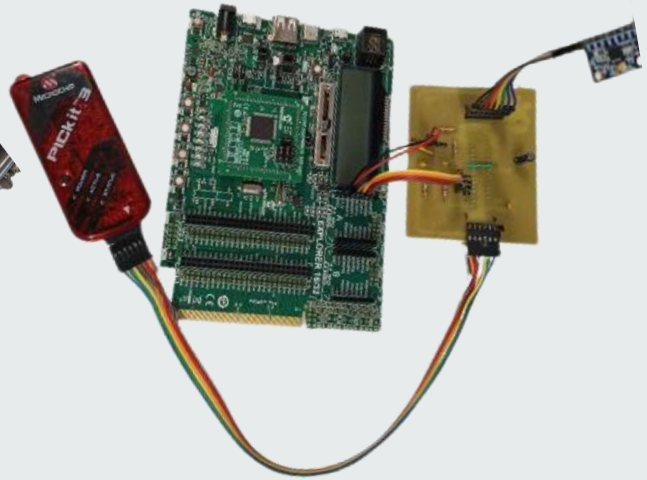
TÉCNICO LISBOA

Oscilloscope

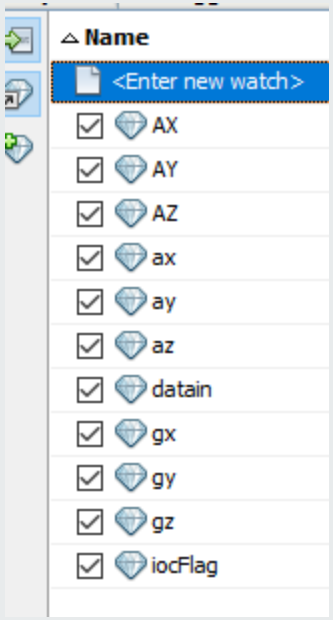
LCD







Debug



TÉCNICO LISBOA

Conclusion...