Worst-Case Execution Time Analysis: computation of the Pitch and Roll angles of an MPU6050 sensor

STR Laboratory Report

2020-21 Q2
March 12th, 2021

Victor Expósito Griñán
Group 10

Contents

Introduction

The objective of this laboratory session of STR is to introduce a critical part when developing Real-Time Systems, and that is the analysis of the computing time of a task. When talking about this computing time, it is mandatory to talk about the Worst-Case Execution Time of the task. In this report I am going to explain how I calculated the WCET of a particular task that computes the Pitch and Roll angles (basic in navigation) of an MPU6050 Inertial Measurement Unit.

Hardware and Software Resources

In order to obtain the wanted results, I needed some specific hardware and software material. Obviously I used the MPU6050 sensor, connected to an Arduino Mega 2560 board (see Fig. 1). To interact with it I used the Arduino IDE, in which an MPU6050 ZIP library was included.

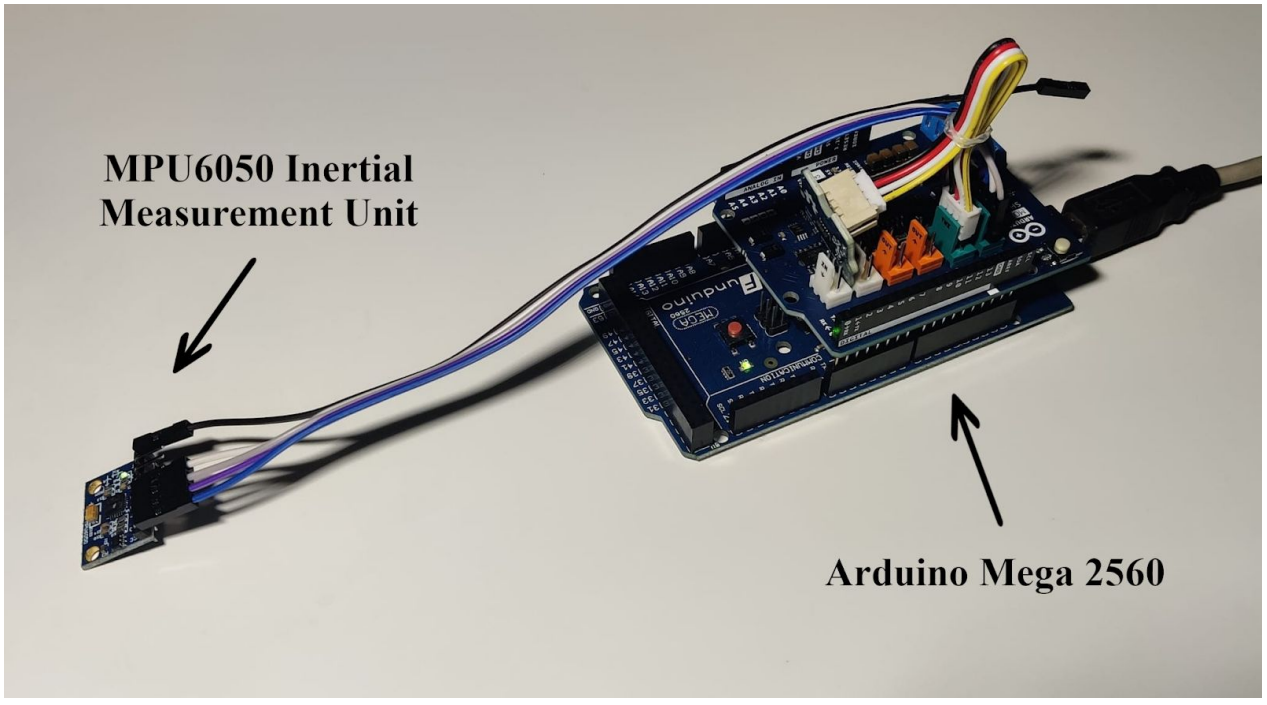

Figure 1: Hardware resources

Code Implementation and Debugging Process

After investigating about the MPU6050 and how to interact with it via code, I proceeded to find information about the Pitch and Roll angles calculation, and finally arrived to a solution that had the following implementation:

```
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
#include "math.h"

MPU6050 MPU6050_sensor;
int16_t datax, datay, dataz;
int16_t pitch, roll;
float Gx, Gy, Gz;

void setup() {
    Serial.begin(38400);
    Wire.begin();
    MPU6050_sensor.initialize();
}

void loop() {
    MPU6050_sensor.getAcceleration(&datax, &datay, &dataz);
    Gx = datax / 16384.0;
    Gy = datay / 16384.0;
    Gz = dataz / 16384.0;
    pitch = atan2(-Gx, sqrt(Gy * Gy + Gz * Gz)) * 180.0 / M_PI;
    roll = atan2(Gy, Gz) * 180.0 / M_PI;
}
```

Figure 2: Code implementation

To check the correctness of the code, I needed a way to debug it, so the simplest and most intuitive one was to represent the information in a temporal graph using the Serial Plotter tool provided by the Arduino IDE. Depending on the angle of the sensor, the graph will print the value in degrees for Pitch (values between -90 and 90) and Roll (values between -180 and 180). For a better visual representation of the data, a small delay (100 ms) was introduced between every sample.

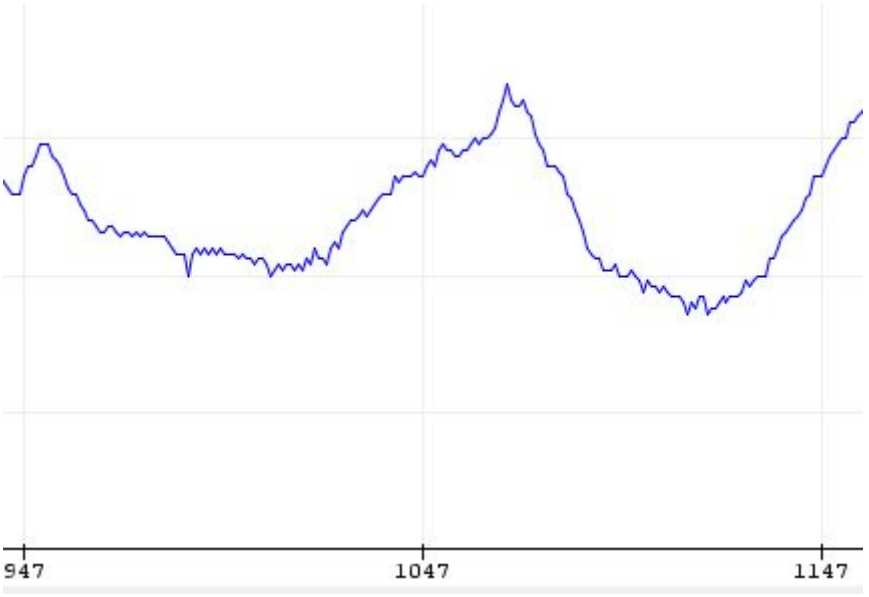

Figure 3: Extract of the Pitch graph

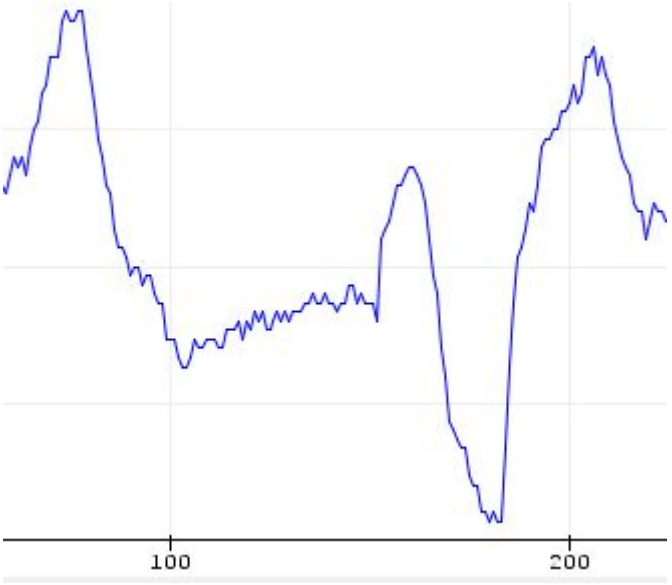

Figure 4: Extract of the Roll graph

WCET calculation

Finally, in order to determine the Worst-Case Execution Time of the explained task, I needed to know how long it takes to make the computation. To get a significant value I left the code running for about a minute, moving the sensor around a little bit, and stored the maximum value of computation in a variable, like this:

```
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
#include "math.h"

MPU6050 MPU6050_sensor;
int16_t datax, datay, dataz;
int16_t pitch, roll;
float Gx, Gy, Gz;
unsigned long startTime, endTime, elapsedTime, WCET = 0;

void setup() {
    Serial.begin(38400);
    Wire.begin();
    MPU6050_sensor.initialize();
}

void loop() {
    startTime = micros();
    MPU6050_sensor.getAcceleration(&datax, &datay, &dataz);
    Gx = datax / 16384.0;
    Gy = datay / 16384.0;
    Gz = dataz / 16384.0;
    pitch = atan2(-Gx, sqrt(Gy * Gy + Gz * Gz)) * 180.0 / M_PI;
    roll = atan2(Gy, Gz) * 180.0 / M_PI;
    endTime = micros();
    elapsedTime = endTime - startTime;
    if (elapsedTime > WCET) WCET = elapsedTime;
    Serial.println(WCET);
}
```

Figure 5: 2nd code implementation

Results

After 90 seconds of printing, I obtained a WCET value of 996 microseconds (0,996 milliseconds). The reason why I used the *micros()* function instead of the *millis()* one was to gain precision in the measurements.

So, as a conclusion, the Worst-Case Execution Time for the computation of the Pitch and Roll angles of an MPU6050 IMU is approximately 1 ms.