



**Academia de Ensino  
Superior de Mafra**

Cursos Técnicos Superiores Profissionais (CTESP)



## ***IoT* COMO DISPOSITIVO MÉDICO**

**BASEADO EM ESP32 USANDO SENSORES MAX30100 E MPU6050**



**RICARDO MIGUEL CURADO MILAGRE Nº81750**

**PROJECTO FINAL**

**ENG. FERNANDO BARROS**

ACADEMIA DE ENSINO SUPERIOR DE MAFRA, 3 de Março de 2023

# ÍNDICE

1 - Introdução .....	2
2 - Descrição do problema .....	1
3 - Desenvolvimento .....	2
3.1. Definição da ideia.....	2
3.2. Desenvolvimento.....	4
3.2.1. Escolha de hardware, aquisição E CUSTOS.....	4
3.2.2. Programas usados e metodologia .....	8
3.2.3. Montagem.....	10
3.2.4. Configuração WAMP64.....	12
3.2.5. Desenvolvimento acelerómetro/giroscópio .....	16
3.2.6. Desenvolvimento pulsímetro / saturação oxigénio .....	26
3.2.7. PHP MPU6050 & MAX30100 .....	33
4 - MS Project Pro .....	35
5 - Resultados .....	37
6 - Conclusões .....	38
7 - Anexos .....	39
7.1. Relatório semanal 16/01 – 27/01 .....	39
7.2. Relatório semanal 30/01 – 06/02.....	40
7.3. Relatório semanal 06/02 – 15/02.....	45
7.4. Relatório semanal 13/02 – 27/02.....	48

## ÍNDICE DE FIGURAS

Figura 1 - Pirâmide demográfica Portuguesa (2016) .....	2
Figura 2 - ESP32 .....	4
Figura 3- M5Stack MAX30100 .....	5
Figura 4- MPU6050 .....	6
Figura 5- MPU6050 montado .....	10
Figura 6 - MAX30100 montado.....	11
Figura 7- WAMP64.....	12
Figura 8- WAMP64 Apache vhosts.conf .....	13
Figura 9- WAMP64 Apache vhosts.conf editado .....	14
Figura 10 - MySQL users.....	14
Figura 11- MySQL tabelas.....	15
Figura 12- Exemplo MPU6050 .....	16
Figura 13 - Excerto de código da primeira versão MPU6050.....	17
Figura 14 - Excerto de código da versão final MPU6050.....	19
Figura 15 - Excerto de código da versão final MPU6050 (2) .....	21
Figura 16 - Código versão final MPU6050.....	22
Figura 17 - Código versão final MPU6050(2) .....	23
Figura 18 - Código versão final MPU6050 (3) .....	24
Figura 19 - Código final MAX30100.....	27
Figura 20 - Código final MAX30100 (2) .....	28
Figura 21 - Código sendMessage do MAX30100.....	29
Figura 22 - Código loop MAX30100.....	30
Figura 23 - Código loop MAX30100 (2) .....	32
Figura 24 - Código PHP MAX30100.....	33
Figura 25 - Código PHP MPU6050 .....	34
Figura 26 - MS Project Pro .....	35
Figura 27- MS Project Pro 16/01 - 27/01 .....	39
Figura 28 - MS Project Pro 30/01 - 06/02 .....	40
Figura 29 - MS Project Pro 30/01 - 06/02 (2) .....	41
Figura 30 - MS Project Pro 06/02 -15/02 .....	45
Figura 31 - MS Project Pro 06/02 -15/02 (2) .....	46
Figura 32- MS Project Pro 13/02 - 27/02 .....	48

## 1 - INTRODUÇÃO

O presente trabalho serviu para realizar um ‘*proof of concept*’ para um dispositivo médico baseado no microcontrolador ESP32, capaz de efectuar leituras de pulsação e saturação de oxigénio através do chip MAX30100, bem como detectar o movimento que a pessoas está a efectuar através do chip MPU6050, detectando situações de queda do utilizador.

Estes dados são depois armazenados em bases de dados para serem posteriormente trabalhados de forma a criar um perfil individual de todos os seus utilizadores de forma a moldar-se as necessidades médicas de cada individuo.

Ao longo deste relatório vamos seguir os métodos utilizados, bem como os processos de montagem, instalação e configuração de software necessário, para chegarmos à conclusão que não só é possível ter um microcontrolador IoT como dispositivo médico, como é necessário que tal aconteça, caso queiramos assegurar cuidados de saúde com qualidade a toda a população não só de Portugal, mas mundial.

## **2 - DESCRIÇÃO DO PROBLEMA**

A necessidade do desenvolvimento deste trabalho deve-se ao facto de estarmos a atravessar por um envelhecimento geral da população mundial. Através do estudo de pirâmides demográficas, conseguimos perceber que a nível global, não está a haver uma substituição nas camadas mais jovens face às mais idosas. Este problema, para além da pressão económica que coloca nos países, irá eventualmente levar a uma falta de mão de obra na área da saúde, uma vez que não é possível haver pessoal qualificado em quantidade para cuidar de tantas pessoas. É por isso necessário a integração cada vez mais de soluções tecnológicas que permitam não descurar os cuidados médicos necessários.

O dispositivo criado permite monitorizar pacientes e familiares que possam ou não viver sozinhos, ou que podem encontrar-se em situações de fraca mobilidade, levando a alguma ansiedade e preocupação por parte dos seus familiares.

Ao ter um dispositivo médico capaz de fornecer dados essenciais sobre uma secção da população vulnerável, conseguimos assegurar o estado de saúde do mesmo ao informar cuidadores imediatamente caso alguma situação aconteça e mais importante ainda, conseguimos transmitir tranquilidade aos familiares e ao próprio utilizador do dispositivo para realizar as suas tarefas no dia a dia, sabendo que, caso algo aconteça, um alerta será emitido.

### 3 - DESENVOLVIMENTO

#### 3.1. DEFINIÇÃO DA IDEIA

Para a disciplina de Projecto Final, foi dada liberdade aos alunos para desenvolver uma ideia própria, que fosse ambiciosa e que utilizasse a maioria dos conhecimentos que foram transmitidos nos últimos 2 anos de forma a poder criar uma solução para um problema específico.

Considerando o envelhecimento da população global, é claro que um dos maiores desafios futuros, vai ser assegurar cuidados de saúde básicos a todas as pessoas que já se encontram, ou caminham, para a terceira idade. Neste momento a idade média em Portugal é de 44 anos (Wikipédia, 2023), e olhando para a figura 1 que representa a pirâmide demográfica portuguesa, é fácil constatar que daqui a uma década, vamos ter um aumento substancial de pessoas que vão necessitar de cuidados médicos.

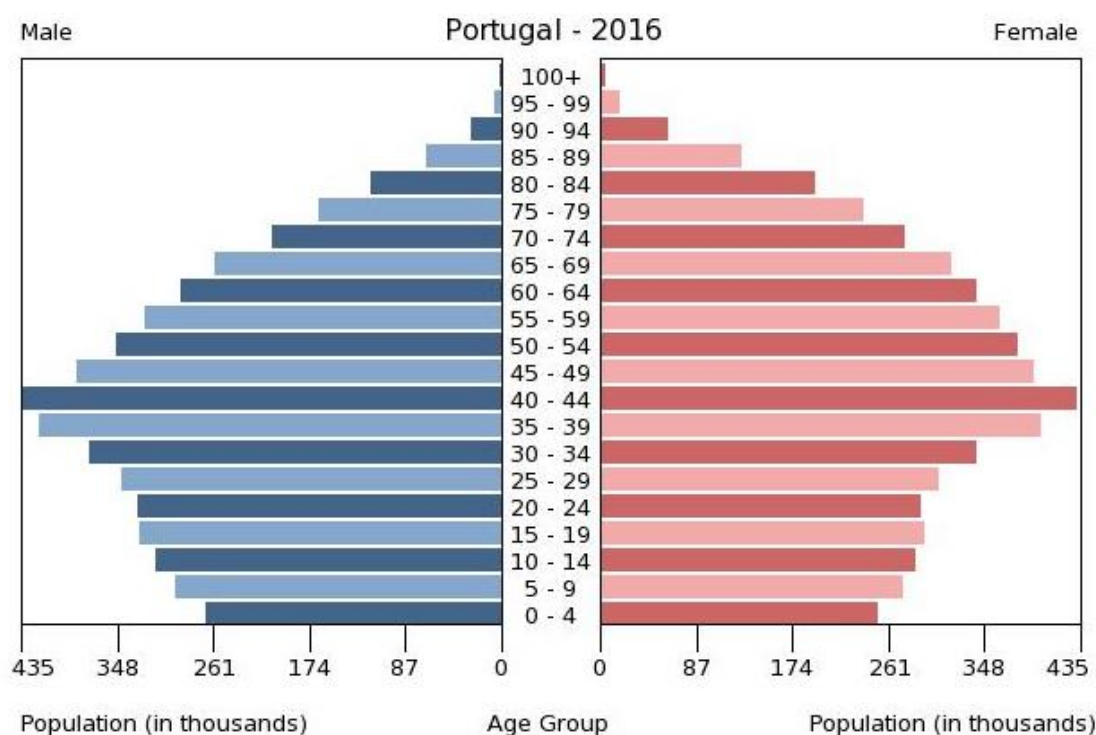


Figura 1 - Pirâmide demográfica Portuguesa (2016)

Observando a pirâmide e prestando mais atenção à disparidade entre quantidade de pessoas entre os seus 35-50 anos e os 15-30 anos é possível chegar à conclusão que vai haver um ponto em que a falta de mão de obra qualificada na área da saúde vai traduzir-se numa necessidade de criação de soluções que tragam os mesmos cuidados de saúde, mas com menos meios humanos. É por isso necessário a criação e desenvolvimento de software capaz de retirar o fardo de algumas tarefas básicas, dando mais tempo livre quer a profissionais de saúde, quer aos próprios utentes.

Tendo em conta estes dados, decidi iria criar um dispositivo que tivesse o seu maior foco de utilização em lares e/ou casas de repouso para idosos. Nestes ambientes, *check-ups* diários aos utentes como verificação de tensão arterial, pulsação e saber se durante a noite houve algum incidente é um processo demoroso e que se traduz num custo humano e monetário desnecessário.

A ideia final apresentada foi a de um dispositivo médico que guardasse numa DB todos os dados dos utentes de um lar/casa de repouso. Estes dados podiam ser depois trabalhados, de forma a construírem um modelo individual que com base em leituras diárias de pulsação, saturação de oxigénio bem como movimentos, permitiam criar um perfil único, calculando médias e desvios, inclusive alertando caso certas medidas não se encontrem dentro da norma, levando à prevenção de certas complicações. Este dispositivo envia também dados para os familiares, contribuindo para uma maior transparência nos cuidados de saúde e tranquilizando os utentes.

## 3.2. DESENVOLVIMENTO

### 3.2.1. ESCOLHA DE HARDWARE, AQUISIÇÃO E CUSTOS

Para o desenvolvimento da ideia, foi necessário perceber quais seriam os dispositivos a ser usados para 'proof of concept'. Durante os 2 anos de curso, tivemos bastante contacto com a *Internet of Things* (IoT), desenvolvimento de bases de dados com a utilização de *SQL Server* e programação em C, C#.

Em IoT, o desenvolvimento foi principalmente com a utilização do ESP8266. Este microcontrolador, embora bom, tem algumas limitações no que diz respeito a conectividade, uma vez que não possui *Bluetooth* e também de rapidez, uma vez que possui uma frequência de 80MHz. Por último, embora seja mais utilizado que o seu irmão ESP32, não tem tanto suporte uma vez que a maioria das bibliotecas de desenvolvimento são adaptadas do ESP32 para o ESP8266 e não ao contrário. Sendo assim, decidi usar o ESP32, que embora tenha um custo mais elevado em cerca de 50% face ao ESP8266, assegura a existência de bibliotecas com o máximo de compatibilidade.



Figura 2 - ESP32



Para leitura de oxigénio no sangue e pulsação, recorri ao chip MAX30100. Infelizmente, não encontrei à venda apenas o chip, e dado as limitações a nível de tempo, foi necessário utilizar o M5Stack MX30100. O chip funciona com a mesma biblioteca, mas possui outras *built-in* para ser utilizado com os restantes chips e acessórios da M5Stack. Para o efeito que se pretende funciona, mas a nível do custo foi bastante mais eleva que o chip normal.



*Figura 3- M5Stack MAX30100*



Estes 2 chips conectam-se ao ESP32 através de portas I2C já predefinidas. São elas o pin D22 e D21. Cada ESP32 tem apenas 2 portas deste tipo, sendo que é possível alterar o código base dentro do ESP32 para ele simular mais 2 portas I2C noutros pin à escolha. Embora seja possível, não é aconselhado fazer, devido não só à complexidade do mesmo, mas também por o facto de ser necessário mexer no código base do ESP32. Outra razão para não o fazer é o preço baixo dos *multiplexer* de portas I2C, que uma vez ligados conseguem simular através de bibliotecas próprias 4, 8 ou 16 portas I2C dependendo do modelo adquirido.

Infelizmente, devido a questões relacionadas com datas de entrega, não foi possível adquirir estes *multiplexers*. A solução encontrada foi a aquisição de outro ESP32.

Antes de apresentar os custos totais deste projecto, é preciso salientar que numa solução final otimizada, em que apenas adquirimos os módulos necessários e os microcontroladores com as funções essenciais, comprados directamente ao produtor com aquisições em lote, os custos de aquisição seriam bastante mais baixos, ou seja, o valor apresentado não corresponde ao custo de uma solução comercial.

Cada ESP32 teve o custo de 17.55€, o *M5Stack* MAX30100 teve um custo de 14.27€ e o MPU6050 teve um custo de 6.09€. A somar a isto temos os custos da *breadboard* dos cabos que são 11.99€.

O custo total para o desenvolvimento deste projecto foi de 67.45€.

### 3.2.2. PROGRAMAS USADOS E METODOLOGIA

Para o desenvolvimento do código para este projecto, foi utilizado o programa *Arduino IDE*, V1.8.19. Este programa, apresenta algumas limitações a nível de desenvolvimento de código, uma vez que é demasiado básico e despido de quaisquer métodos que ajudam os utilizadores a encontrar possíveis erros. Nem um *debugger* a versão utilizada possui. Existe uma versão mais recente (V2.0.4) mas tem bastantes erros de ligação aos ESP32 que foram visíveis durante as primeiras tentativas de ligação efectuadas. A escolha deste IDE em detrimento de um ‘*Visual Studio*’ deveu-se meramente ao recurso tempo, que não abundava, e preferi utilizar uma ferramenta que já dominava.

A metodologia utilizada neste trabalho foi decidida logo ao início. O foco estaria no desenvolvimento de código, alocando grande parte do tempo para a sua criação e optimização, utilizando depois o tempo restante para a elaboração deste relatório. Esta decisão deveu-se a, principalmente, dois factores.

O primeiro é o facto do ESP32 não ser capaz de comunicar directamente com uma BD. Existem bibliotecas disponíveis que o fazem, mas estas são de terceiros e carecem de actualizações, portanto não foram consideradas para o trabalho. O objectivo é simplesmente usar os recursos que o ESP32 e os chips tem, de forma ao programa poder ser sempre executado, sem depender de software de terceiros. Uma potencial actualização do *firmware* do ESP32, ou das próprias DB podia alterar as bibliotecas, levando a incompatibilidades temporárias, assumindo que as bibliotecas continuam a ter suporte. No caso de um dispositivo médico, não podemos correr esse risco, como tal, apenas foram utilizadas as bibliotecas oficiais oferecidas por os fabricantes.

O problema ao tomar esta decisão, é que o trabalho ganha complexidade. Essa complexidade passa por utilizar mais código na ligação à DB e mais programas de forma a assegurar a conexão.

O segundo factor é a optimização. Um código tem de ser funcional, mas sobretudo eficiente, o desenvolvimento e testes não podem ser algo que não a primeira prioridade. Tem de funcionar sempre e de maneira a usar menos recursos possível. Esta necessidade é ainda maior, quando temos dispositivos com processadores de 240MHz. A falta de optimização nota-se mais aqui do que em qualquer outro dispositivo. Para além disso, se o objectivo é ser um projecto comercializável, quanto menos recursos usarmos melhor. Menos custo terá o produto final aumentando a rentabilidade económica do mesmo.

Para a simulação da DB onde os dados dos utentes seriam guardados, usamos o programa WAMP64 a correr em máquina local, em detrimento de uma solução *web hosting*. Esta escolha deveu-se à questão financeira, uma vez que seria necessário alugar o domínio por um período de 1 ano, e devido à sua semelhança com uma DB real.

O WAMP64 é um acrónimo para os programas que ele próprio engloba. Ou seja, *Windows*, *Apache*, *MySQL* e *PhP*. Iremos utilizar todos na elaboração deste trabalho. O *PhP* será o que permite a comunicação com *MySQL* e o Apache servirá para virtualizar o ficheiro *PhP* de forma a este poder ser acedido por o ESP32.

### 3.2.3. MONTAGEM

Para a montagem, foi utilizada uma *breadboard*, onde foram depois inseridos os 2 chips, cada um deles ligado com a porta SDA ao pin D21 e a porta SDL ao pin D22. Como já foi mencionado, estas ligações já estão predefinidas no ESP32, ou seja, não é necessário indicar o pin que está a ser utilizado por os chips. O MPU6050 foi ligado aos 3V fornecidos por o ESP32, enquanto o MAX30100 necessitou de um input de 5V de uma *powerbank*, com um cabo USB adaptado.

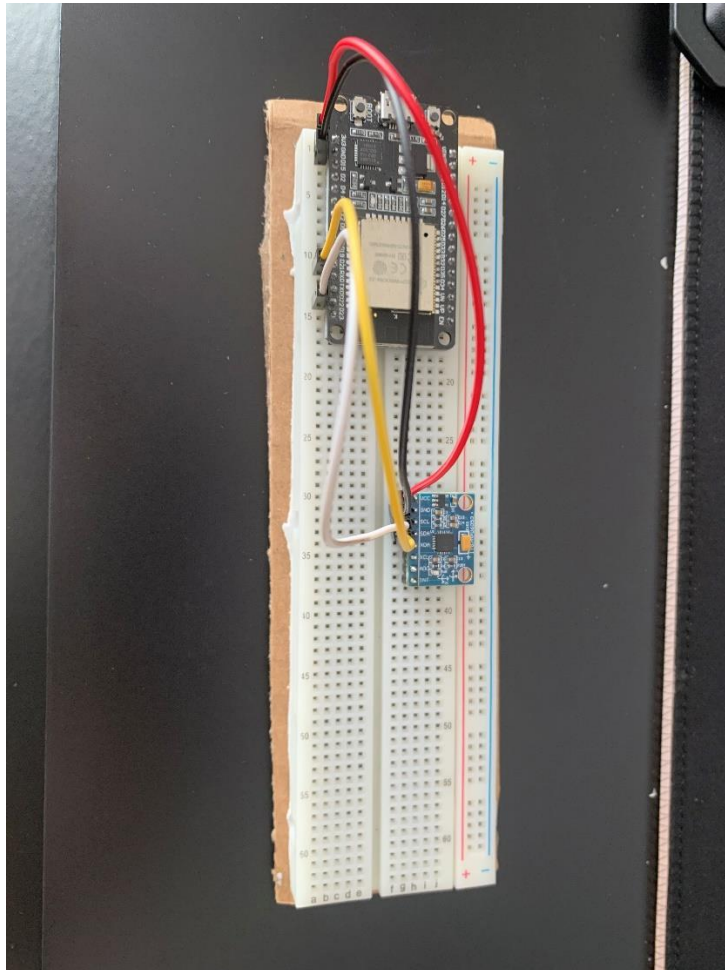
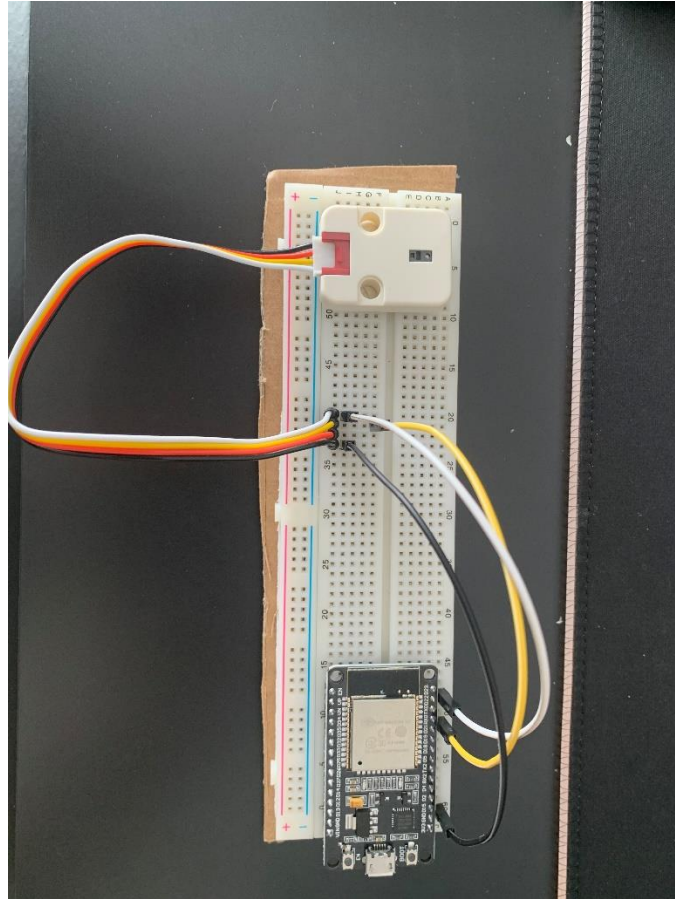


Figura 5- MPU6050 montado

A razão por a qual o MAX30100 utiliza a *powerbank* e não o ESP32 é devido à sua necessidade de ser alimentado por 5V e não 3.3V. Este chip tem efectivamente 2 sistemas que requerem alimentações diferentes. O LED presente necessário para efectuar a leitura com maior certeza, é alimentado com uma voltagem mínima de 3.1V (a intensidade do LED pode ser controlada) e o sistema de leitura requer 1.7V.



*Figura 6 - MAX30100 montado*

### 3.2.4. CONFIGURAÇÃO WAMP64

O WAMP64 é um programa que agrega um conjunto de ferramentas importantes e que foi utilizado como base para a realização deste trabalho. As configurações necessárias, embora sejam poucas, são muito específicas e a documentação existente é quase nula, levando a bastante uso de fóruns online como por exemplo o *stackoverflow* para perceber as melhores definições a aplicar.

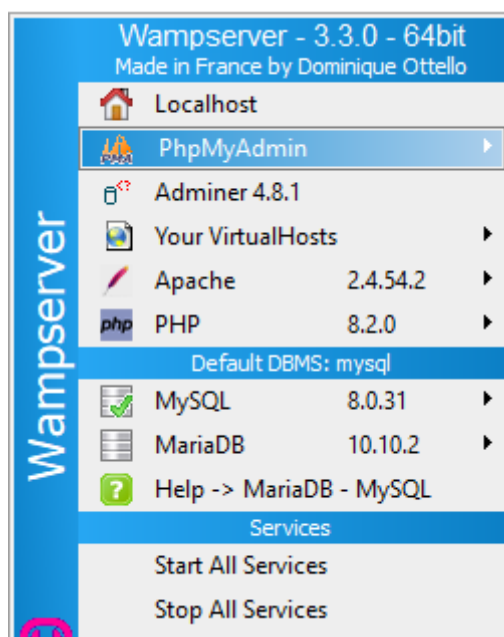


Figura 7- WAMP64

Um aspecto que foi necessário alterar, foi a possibilidade de virtualização de pasta onde se encontrava os ficheiros *Php* dos 2 diferentes ESP32. Por definição e por uma questão de segurança, esta opção vem desactivada no Apache, ou seja, é necessário colocar a pasta em ambiente de virtualização.



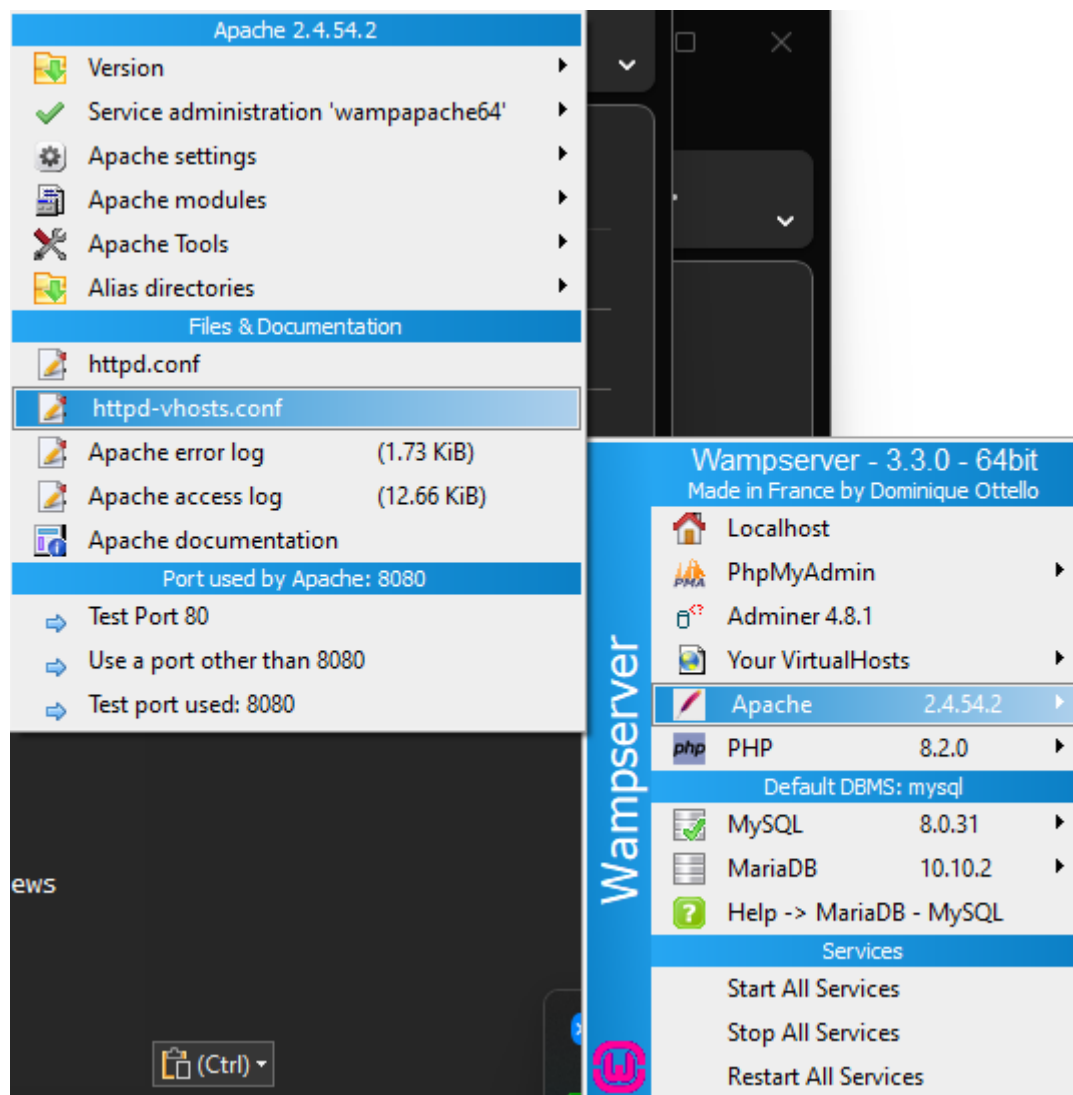


Figura 8- WAMP64 Apache vhosts.conf

Foi, portanto, necessário colocar a *root* para a pasta que pretendemos virtualizar. No meu caso, o pretendido era virtualizar a pasta 'www', uma vez que era lá que se encontrava os ficheiros *PhpP*.

```

<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot "C:\wamp\www"
    ServerName localhost
    ErrorLog "logs/localhost-error.log"
    CustomLog "logs/localhost-access.log" common
</VirtualHost>

<VirtualHost *:80>
    ServerName www.testZendTutorial.com
    DocumentRoot "C:\wamp\www\connect.php\public"
    SetEnv APPLICATION_ENV "development"
    <directory "C:\wamp\www\connect.php\public">
        options +Indexes +FollowSymLinks +MultiViews
        AllowOverride All
        Require local
        Require ip 192.168.0.0/16
    </directory>
</VirtualHost>

```

Figura 9- WAMP64 Apache vhosts.conf editado

Para além de pastas, é possível atribuir a um ficheiro específico um *servername*. Este nome pode depois ser introduzido no browser para ter acesso ao conteúdo.

Dentro do *PHP*, na secção *MySQL*, foi também necessário a criação de utilizadores para os 2 ESP32. Um com o nome *esp32max* e o outro *esp32mpu*. A razão por a qual não usam o mesmo *username* é devido ao erro de conexão que iria certamente ocorrer. Para além disso, caso todos utilizassem o mesmo *login*, não era possível aceder à tabela em tempo real e ver os dados que seriam injectados. Ao dar permissões totais a ambos os utilizadores, certificamo-nos que conseguem aceder sem problemas. Na versão final, as permissões teriam de ser editadas de forma aos ESP32 poderem ler e acrescentar dados, sem a possibilidade de os apagar e manipular.

## User accounts overview




	User name	Host name	Password	Global privileges	Grant	Action
<input type="checkbox"/>	esp32max	%	No	ALL PRIVILEGES	Yes	 Edit privileges  Export  Lock
<input type="checkbox"/>	esp32mpu	%	No	ALL PRIVILEGES	Yes	 Edit privileges  Export  Lock

Figura 10 - MySQL users

Por último, foram criadas 2 tabelas, que vão guardar os dados obtidos. Todos os dados são *float*, excepto o estado, que é inserido como *varchar*, *timestamp* que é automático sempre que um valor é adicionado à tabela e uma *key* que é auto incrementada.

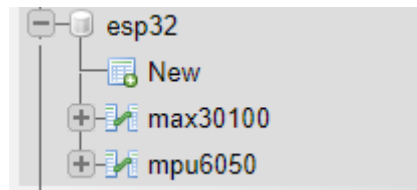


Figura 11- MySQL tabelas

### 3.2.5. DESENVOLVIMENTO ACCELERÓMETRO/GIROSCÓPIO

O código para desenvolvimento das medições para o acelerómetro/giroscópio, foi o mais simples de perceber, uma vez que o próprio chip, traz como exemplo na biblioteca como obter medições através da leitura.

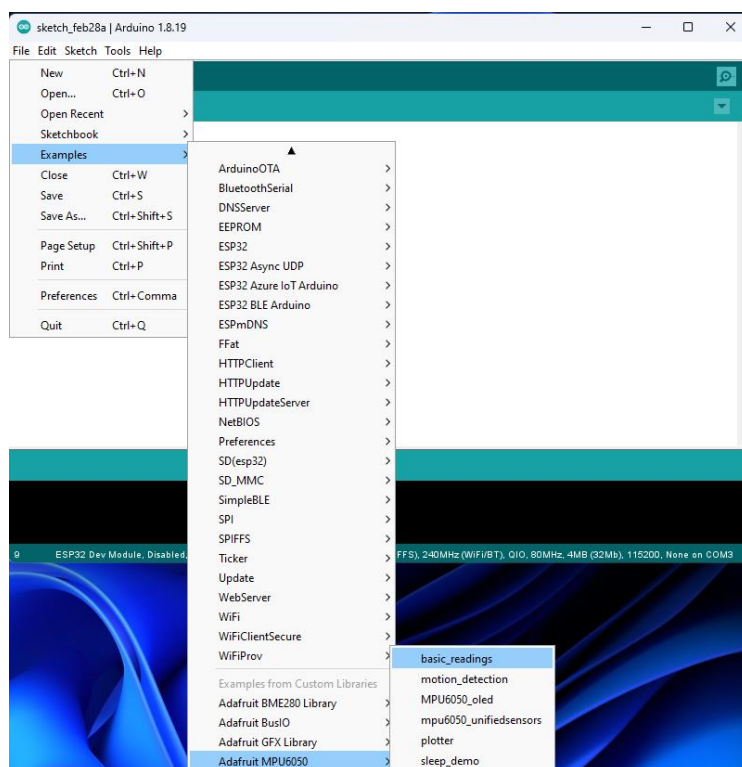


Figura 12- Exemplo MPU6050

Com este código básico fornecido, foi possível concluir que o chip carece de calibração para o *threshold* desejado quer a nível de aceleração, quer a nível de giroscópio. Após inicialização e uma vez em *loop*, o MPU6050 faz as suas leituras após um período definido por o utilizador, volta a efectuar leituras.

Após algumas mudanças, obtivemos um código que fazia as leituras, mas ainda com demasiados passos incorrectos.

```
void loop()
{
    // Lê o sensor
    sensors_event_t a, g, temp;
    mpu.getEvent(&a, &g, &temp);

    float ax = a.acceleration.x * 9.81;
    float ay = a.acceleration.y * 9.81;
    float az = a.acceleration.z * 9.81;

    // magnitude da aceleração
    float magnitude = sqrt(ax*ax + ay*ay + az*az);

    // Se a magnitude é proxima de 9.8, a pessoas esta de pe
    if (abs(magnitude - 9.8) < 0.1)
    {
        if (!wasStanding) {
            Serial.println("Levantado");
            wasStanding = true;
        }
    }
    // caso contrario, se estiver proxima de 0, sentado
    else if (abs(magnitude) < 0.1) {
        if (wasStanding) {
            Serial.println("Sentado");
            wasStanding = false;
        }
    }
    // andar
    else {
        if (az > 0.3) {
            stepCount++;
            Serial.print("Step count: ");
            Serial.println(stepCount);
        }
    }

    delay(500);
}
```

Figura 13 - Excerto de código da primeira versão MPU6050

Os erros aqui cometidos foram resolvidos com uma maior compreensão do código e das capacidades do MPU6050. O chip interpreta sempre o valor da gravidade, ou seja, dependendo da posição, pode exercer a sua força que é constante no vector x, y ou z. Com o MPU6050 na posição horizontal, podemos sempre compensar o valor da gravidade no eixo z, ao retirar 9.8 metros/segundo<sup>2</sup> ao valor medido. Multiplicar, como está demonstrado na figura, o valor por 9.8 metros/segundo<sup>2</sup> não faz nada a não ser aumentar ainda mais as leituras e a incongruência das mesmas, uma vez que mesmo multiplicando e ajustando os valores, o eixo z ou eixo x ou eixo y, dependendo da orientação, manter-se-ia sempre com 9.8 metros/segundo<sup>2</sup> a mais na leitura.

Outro erro nesta primeira abordagem é a tentativa de construção de um modelo para saber se o utilizador está de pé ou se está sentado. Esta abordagem é incorrecta uma vez que basta o utilizador saltar, para apresentar a indicação de sentado, quando não é o caso.

Ao longo da elaboração deste trabalho, ficou claro que não era possível, dada as limitações do ESP32, trabalhar os dados obtido a não ser de forma muito rudimentar. Na elaboração dos códigos deste ponto para a frente, assumimos que o chip se iria encontrar na posição horizontal e que a força da gravidade afectaria apenas o eixo z.

```
// BIBLIOTECAS
#include <Wire.h>
#include <WiFi.h>
#include <Adafruit_MPU6050.h>
#include <UrlEncode.h>
#include <HTTPClient.h>

// DEFINIÇÕES
Adafruit_MPU6050 mpu;
HTTPClient http;S

// DEFINIÇÃO DOS LIMITE E INICIO DO ESTADO
float accel_limite = 10.0; // LIMITE PARA ACCELL
float gyro_limite = 2.0; // LIMITE PARA GYRO
float andar_limite = 2.0; // LIMITE PARA ANDAR
String estado = "";

// DEFINIÇÕES WIFI
const char* ssid = "C-137";
const char* password = "18e457b313";
const char* host = "192.168.1.72";
WiFiClient client;

//DEFINIÇÕES P/CALLME BOT
String phoneNumber = "+351963151007";
String apiKey = "7129350";

// INICIAR
void setup()
{
    // CONECTAR SERIAL
    delay(1500);
    Serial.print ("A iniciar serial");
    Serial.begin(115200);
    Wire.begin();

    // CONECTAR WIFI
    Serial.println("");
    WiFi.begin(ssid, password);
    Serial.println("A iniciar WiFi");
    while(WiFi.status() != WL_CONNECTED)
    {
        delay(250);
        Serial.print(".");
    }
    if (WiFi.status() == WL_CONNECTED)
    {
        Serial.println("");
        Serial.print("Conectado com Sucesso!");
        Serial.println("IP: ");
        Serial.println(WiFi.localIP());
        Serial.println("");
    }
    else
    {
        Serial.println("");
        Serial.print("Falha a inciar WiFi");
    }
}
```

---

Figura 14 - Excerto de código da versão final MPU6050

Na versão final, começamos por declarar as bibliotecas necessárias para o funcionamento do programa. Para além das bibliotecas WiFi e MPU6050, É necessário também a utilização da biblioteca *HTTPClient* e *UrlEncode*.

Estas duas últimas, vão ser as usadas fazer a comunicação entre o sistema de alertas por *WhatsApp* bem como o sistema de alerta do *Callmebot*.

Definimos o objecto MPU6050 como mpu e o objecto *HTTPClient* como http. De seguida, inserimos um *threshold* para uma aceleração, giroscópio e andar serem limites, ou seja, o valor a partir da qual uma aceleração e movimento angular passa do limite que nós consideramos normal e o momento a partir da qual, a aceleração indica que o MPU6050 se encontra a andar. Por último definimos a variável estado.

Após definirmos credenciais da internet e do *host* ao qual queremos transmitir as nossas leituras, bem como o telefone (ou telefones) com a respectiva API para a qual queremos comunicar um evento, iniciamos o *setup*.

Vamos iniciar o serial em 115200. O ESP32 funciona mais correctamente dentro deste serial, podendo haver perda/alteração de dados caso estejamos em *serials* mais fracos. De seguida, iniciamos a ligação *wifi*. Em todos estes casos, bem como nos restantes existem timers e impressão em serial para uma maior compreensão por parte do utilizador. No produto final estes timers seriam abolidos para privilegiar a rapidez de ligação e de medições.



```
// INICIAR MPU
mpu.begin();

Serial.println("A iniciar MPU6050");
delay(1000);
if (!mpu.begin())
{
    Serial.println("");
    Serial.println("Falha ao iniciar MPU6050");
    while (1)
    {
        delay(10);
    }
}

// DEFINIÇÕES DE RANGE
mpu.setAccelerometerRange(MPU6050_RANGE_16_G);
mpu.setGyroRange(MPU6050_RANGE_2000_DEG);
}

// CÓDIGO EDITADO PARA USO DO CALLMEBOT.
// CÓDIGO ORIGINAL DISPONÍVEL EM CALLMEBOT.COM
void sendMessage(String message)
{
    // Data to send with HTTP POST
    String url = "https://api.callmebot.com/whatsapp.php?phone=" + phoneNumber + "&apikey=" + apiKey + "&text=" + urlEncode(message);
    HTTPClient http;
    http.begin(url);

    // Specify content-type header
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");

    // Send HTTP POST request
    int httpResponseCode = http.POST(url);
    if (httpResponseCode == 200){
        Serial.print("Mensagem enviada com sucesso");
        Serial.println("Retomando leituras");
    }
    else{
        Serial.println("Erro ao enviar mensagem");
        Serial.print("HTTP response code: ");
        Serial.println(httpResponseCode);
    }

    // Free resources
    http.end();
}
```

Figura 15 - Excerto de código da versão final MPU6050 (2)

Após iniciarmos o mpu, chegamos ao código do *Callmebot*. Este código é da auditoria do website callmebot.com e permite uma simples ligação https com um pedido para envio de mensagem estabelecida por o utilizador. Embora o código não seja da minha autoria, é praticamente igual à forma como vamos comunicar com o WAMP64 para inserir os dados em DB.

```

void loop()
{
    // DEFINIÇÃO DE ACORDO COM MPU6050
    sensors_event_t accel, gyro, temp; // TEMP NÃO USADO
    mpu.getEvent(&accel, &gyro, &temp); //TEMP NÃO USADO

    // CALCULO DAS MAGNITUDES
    // MAGNITUDE ACELERAÇÃO - RETIRAMOS 9.8, UMA VEZ QUE É O VALOR DA ACELE-
    RAÇÃO GRAVITICA
    float accel_magnitude = (sqrt(accel.acceleration.x*accel.acceleration.x +
                                   accel.acceleration.y*accel.acceleration.y +
                                   accel.acceleration.z*accel.acceleration.z))-
9.8;

    // MAGNITUDE ANGULO
    float gyro_magnitude = sqrt(gyro.gyro.x*gyro.gyro.x +
                                   gyro.gyro.y*gyro.gyro.y +
                                   gyro.gyro.z*gyro.gyro.z);

    // MAGNITUDE THRESHOLD ANDAR
    float accel_andar = sqrt(accel.acceleration.x*accel.acceleration.x +
                                   accel.acceleration.y*accel.acceleration.y);

    // CALCULO HORIZONTAL/VERTICAL COM CONVERSÃO PARA GRAUS
    float pitch = atan2(accel.acceleration.x, sqrt(accel.acceleration.y*ac-
cel.acceleration.y +
                                   accel.accelera-
tion.z*accel.acceleration.z)) * 180.0 / PI;
    float roll = atan2(accel.acceleration.y, sqrt(accel.acceleration.x*ac-
cel.acceleration.x +
                                   accel.acceleration.z*ac-
cel.acceleration.z)) * 180.0 / PI;

    // CAOS PASSE O THRESHOLD DEFINIDO
    if (accel_magnitude > accel_limite && gyro_magnitude < gyro_limite)
    {
        // CASO PASSE OS LIMITES ESTABELECIDOS PELO USER, EMITE UM ALERTA DE
        QUEDA
        Serial.print("");
        Serial.println("Queda Detectada!");
        Serial.print(" | Horizontal: ");
        Serial.print(pitch);
        Serial.print(" | Vertical: ");
        Serial.print(roll);
        Serial.println("");
        sendMessage("ALERTA: Foi detectada uma queda");
        // DELAY DEVIDO A POSSIVEIS INCONGRUENCIAS DE LEITURAS SEGUINTE
    }

    // SIMPLIFICAÇÃO DE ESTADO PARADO
    if (accel_andar > -1 || accel_andar < 1)
    {
        estado = "parado";
    }

    //SIMPLIFICAÇÃO ESTADO ANDAR
    if (accel_andar > 1 && accel_andar < 3)
    {
        estado = "andar";}

```

Figura 16 - Código versão final MPU6050

```
// SIMPLIFICAÇÃO ESTADO CORRER
if (accel_andar > 3)
{
    estado = "correr";
}

// IMPRESSÃO EM SERIAL PARA VISUALIZAÇÃO
Serial.print("Accel x: ");
Serial.print(accel.acceleration.x);
Serial.print(" | Accel y: ");
Serial.print(accel.acceleration.y);
Serial.print(" | Accel z: ");
Serial.print(accel.acceleration.z);
Serial.print(" | Gyro x: ");
Serial.print(gyro.gyro.x);
Serial.print(" | Gyro y: ");
Serial.print(gyro.gyro.y);
Serial.print(" | Gyro z: ");
Serial.print(gyro.gyro.z);
Serial.print(" | Estado: ");
Serial.print(estado);

// INDICA CONEXÃO
Serial.print("A conectar a ");
Serial.println(host);

// ESTABELECE CONEXÃO COM WIFILIENT
WiFiClient client;
const int httpPort = 8080;
if (!client.connect(host, httpPort)) {
    Serial.println("conexão falhada");
    return;
}

// CONSTRUÇÃO DA STRING DE ENVIO
String url = "/connectmpu?";
url += "acceleration_x=" + String(accel.acceleration.x) + "&";
url += "acceleration_y=" + String(accel.acceleration.y) + "&";
url += "acceleration_z=" + String(accel.acceleration.z) + "&";
url += "gyro_x=" + String(gyro.gyro.x) + "&";
url += "gyro_y=" + String(gyro.gyro.y) + "&";
url += "gyro_z=" + String(gyro.gyro.z) + "&";
url += "estado=" + String(estado);

// GET REQUEST
Serial.println("A enviar pedido HTTP...");
client.print(String("GET ") + url + "HTTP/1.1\r\n" +
    "Host: " + host + "\r\n" +
    "Connection: close\r\n\r\n");

// THRESHOLD TIMEOUT && ESTABELECE RERUN
unsigned long timeout = millis();
while (client.available() == 0) {
    if (millis() - timeout > 1000) {
        Serial.println(">>> Client Timeout !");
        client.stop();
        return;
    }
}
```

Figura 17 - Código versão final MPU6050(2)

```
// DEVOLVER O QUE O SERVER INDICA
while(client.available()) {
    String line = client.readStringUntil('\r');
    Serial.print(line);

}

// FECHAR CONEXÃO
Serial.println();
Serial.println("Conexão fechada");
delay (1000);
}
```

Figura 18 - Código versão final MPU6050 (3)

Neste ciclo *loop*, é efectuada a definição dos sensores e dos eventos que o MPU6050 regista.

De seguida são iniciadas 5 variáveis. A ‘*accel\_magnitude*’ calcula a magnitude máxima da aceleração. O mesmo faz o ‘*gyro\_magnitude*’ mas para o ângulo. A ‘*accel\_andar*’ faz o mesmo que a variável *accel\_magnitude*, mas sem o eixo z, ou seja, sem o eixo que, tal como definimos, exerce a força da gravidade. O que isto permite é saber apenas a magnitude do eixo x, y. Por último *pitch* e *roll* indica a quantidade de movimento horizontal e vertical em ângulos.

Uma vez que os valores estabelecidos no início do código permitem ter um *threshold* a partir do qual consideramos o movimento uma queda, utilizamos um ‘*if*’ para comparar estes com os valores lidos. Caso esses valores sejam superiores, é indicado em serial que houve uma queda e o código *callmebot* é activado com a mensagem “ALERTA: Foi detectada uma queda”.

Ao mesmo tempo e como já foi referido, conseguimos ter uma noção básica se o utilizador está a correr, andar ou parado aquando da queda. Esta parte do código é bastante rudimentar e infelizmente não houve tempo suficiente para implementação de *machine learning* para ler todos os valores e treinar o modelo para obter resultados mais aperfeiçoados. Tal seria possível com um código *python* por exemplo, através do uso das bibliotecas próprias. Desenvolvendo esse módulo, podemos criar modelos para cada um dos utentes do lar, em que criamos padrões próprios e conseguimos saber se o evento catalogado como queda é verdadeiro ou não, analisando o movimento anterior e no movimento a seguir. Neste momento, a implementação realizada é frágil e carece de alguma tolerância uma vez que produz bastantes resultados incorrectos. De seguida, os valores da aceleração, giroscópio e estado são impressos em serial, para uma questão de visualização. No programa final, não existiria a necessidade de imprimir nada.

É estabelecida a ligação entre o ESP32 e a máquina que esta a correr o programa WAMP64. Caso consiga, é criada uma *string* com os valores lidos da aceleração, giroscópio e estado e é feito um pedido à máquina. É feita a população do ficheiro *Php* e o pedido é enviado inserindo os dados em MySQL.

Existe depois um período de *timeout*, ou seja, caso não haja comunicação de volta, o pedido expirou, seguido de uma impressão que é enviada do *MySQL* confirmado ou não a introdução dos dados. A conexão é depois fechada.

É importante referir que existem outras formas de fazer o programa e o pedido à DB. No meu caso, o pedido é feito por o ficheiro *Php*, sem a necessidade de o carregar para o ESP32, no entanto, era possível importar o ficheiro *Php*, popular e enviar como um todo para o WAMP64. Mais uma vez, a decisão de enviar apenas os dados e tratar do pedido na máquina local é por uma questão de optimização. Ao carregar popular e enviar estamos a realizar passos desnecessários que apenas aumentam a complexidade do processo. Já que temos o ficheiro alojado em *virtualhost* por o Apache, é mais fácil enviar os dados e deixar a máquina com a sua maior capacidade de processamento tratar do pedido.

Outro aspecto também importante de voltar a referir prende-se com o facto dos dados em DB poderem ser trabalhados de forma a criarem modelos individuais de cada utilizador. Podemos ter médias para movimentos e mobilidade em ângulos o que permite em conjugação com os restantes dados que iremos adquirir, antecipar algum problema médico que nem o utilizador tenha ainda conhecimento. A melhor saúde que podemos ter é uma saúde preventiva, em que antecipamos o problema, fazendo com que as suas consequências não sejam tão nefastas.

### 3.2.6. DESENVOLVIMENTO PULSÍMETRO / SATURAÇÃO OXIGÉNIO

O desenvolvimento do código do pulsímetro, foi realizado com a adaptação do código do MPU6050. A razão por a qual foi uma adaptação, deve-se ao facto de os 2 chips funcionarem de forma semelhante, a nível de inicialização e leitura. A diferença que existe é apenas com variáveis declaradas e cálculo que são executados para a apresentação de algumas métricas. Sem elas, o código era muito semelhante.

Um dos desafios com este chip foi a coerência nas leituras de pulsação.

Por a pesquisa que efectuei, o chip original MAX30100 não sofre deste problema. Sem ter certezas, assumo que se deve ao facto de estarmos a usar o M5Stack, que pode estar mais calibrado para dar leituras mais fiáveis quando conectado aos restantes acessórios da marca. A nível de definições internas do chip, conseguimos alterar a intensidade do LED, a *sample rate*, e o *FIFO buffer*. Este último, determina quantas leituras o chip tem de efectuar até libertar um valor médio. Mesmo mexendo nestas definições, não foi possível ter leituras sem variações bruscas na pulsação. Como o tempo disponível não foi muito, foi impossível encomendar um chip MAX30100 original, bem como foi impossível comprar os restantes acessórios da marca, uma vez que os preços são bastante elevados.

Os valores obtidos são assim irrelevantes para o *showcase* do hardware, mas permitem-nos concluir que com um dispositivo deste tipo é possível retirar medições de pulsação e saturação de oxigénio de um individuo.

```
//BIBLIOTECA
#include <Wire.h>
#include <WiFi.h>
#include "MAX30100_PulseOximeter.h"
#include <UrlEncode.h>
#include <HttpClient.h>
#define tempoUpdate      1000
#define arrayPulsacao    10

//DEFINIÇÕES
PulseOximeter pox;
HttpClient http;

// DEFINIÇÃO DOS TEMPOS LIMITE E ARRAYS
int ultimaMedicao = 0;
int arraybpm[arrayPulsacao];
int semPulsacao;

// DEFINIÇÕES WIFI
const char* ssid = "C-137";
const char* password = "18e457b313";
const char* host = "192.168.1.72";
WiFiClient client;

//DEFINIÇÕES P/CALLME BOT
String phoneNumber = "+351963151007";
String apiKey = "7129350";

void setup()
{
    // CONECTAR SERIAL
    delay(1500);
    Serial.print ("A iniciar serial");
    Serial.begin(115200);
    Wire.begin();

    // CONECTAR WIFI
    Serial.println("");
    WiFi.begin(ssid, password);
    Serial.println("A iniciar WiFi");
    while(WiFi.status() != WL_CONNECTED)
    {
        delay(250);
        Serial.print(".");
    }
    if (WiFi.status() == WL_CONNECTED)
    {
        Serial.println("");
        Serial.print("Conectado com Sucesso!");
        Serial.println("IP: ");
        Serial.println(WiFi.localIP());
        Serial.println("");
    }
    else
    {
        Serial.println("");
        Serial.print("Falha a inciar WiFi");
    }
}
```

Figura 19 - Código final MAX30100

```
// INICIAR POX
pox.begin();

Serial.println("A iniciar MAX30100");
delay(1000);
if (!pox.begin())
{
    Serial.println("");
    Serial.println("Falha ao iniciar MAX30100");
    while (1)
    {
        delay(10);
    }
}

// CONFIGURAÇÃO LED 8MA
pox.setIRLedCurrent(MAX30100_LED_CURR_46_8MA);

// INICIAR ARRAY PULSAÇÃO, VALORES DEFAULT
for (int i = 0; i < arrayPulsacao; i++)
{
    arraybpm[i] = -1;
}
}
```

Figura 20 - Código final MAX30100 (2)

Como podemos observar, existem bastantes semelhanças entre o código em cima e o do chip MPU6050. As bibliotecas são iguais, com excepção a do chip MAX30100.

Para a realização desta leitura, optei por a criação de um *array* que guardasse os valores dos batimentos cardíacos. Este *array*, vai permitir guardar leituras calculando depois uma media de todas as leituras efectuadas. Também são inicializadas 2 variáveis para controlo caso o utente esteja sem pulsação durante um tempo a definir, e o tempo que passou desde a ultima medição.

De resto o código é igual ao MPU6050, com as definições para o *Callmebot*, *Wifi*. A única diferença no *setup* é a inicialização do *pox* e não do *mpu*, seguido da definição da intensidade do LED. Quanto mais elevado a intensidade, maior a fiabilidade das leituras, ou seja, está no máximo permitido de 8ma.

Por último, iniciamos o *array* da pulsação com um valor *default*.



```
void sendMessage(String message)
{
    // Data to send with HTTP POST
    String url = "https://api.callmebot.com/whatsapp.php?phone=" +
    phoneNumber + "&apikey=" + apiKey + "&text=" + urlEncode(message);
    http.begin(url);

    // Specify content-type header
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");

    // Send HTTP POST request
    int httpStatusCode = http.POST(url);
    if (httpStatusCode == 200){
        Serial.print("Mensagem enviada com sucesso");
    }
    else{
        Serial.println("Erro ao enviar mensagem");
        Serial.print("HTTP response code: ");
        Serial.println(httpStatusCode);
    }

    // Free resources
    http.end();
}
```

*Figura 21 - Código sendMessage do MAX30100*

```
void loop()
{
    // UPDATE VALORES
    pox.update();

    // VERIFICA SE PASSO O TEMPO NECESSÁRIO ENTRE LEITURAS
    if (millis() - ultimaMedicao > tempoUpdate)
    {
        // LE OS VALORES
        int bpm = pox.getHeartRate();
        int spo2 = pox.getSpO2();

        // GUARDA O NOVO VALOR BPM NO FIM DO ARRAY
        if (bpm > 0)
        {
            for (int i = 0; i < arrayPulsacao - 1; i++)
            {
                arraybpm[i] = arraybpm[i + 1];
            }
            arraybpm[arrayPulsacao - 1] = bpm;
        }

        // CALCULA A MEDIA DAS ULTIMAS LEITURAS SE FOR NÃO NULO
        int sum = 0;
        int count = 0;
        for (int i = 0; i < arrayPulsacao; i++) {
            if (arraybpm[i] != -1) {
                sum += arraybpm[i];
                count++;
            }
        }
        float mediabpm = (count > 0) ? (float)sum / count : 0;

        // IMPRESSÃO EM SERIAL PARA VISUALIZAÇÃO
        Serial.print("Heart rate:");
        Serial.print(bpm);
        Serial.print("bpm / SpO2:");
        Serial.print(spo2);
        Serial.print("% / Average Heart Rate:");
        Serial.print(mediabpm);
        Serial.println(" bpm");

        // RESET TIMER SE NÃO FOR 0
        if (bpm > 0)
        {
            semPulsacao = 0;
        }
        else
        {
            // INCREMENTA O VALOR SEM PULSAÇÃO
            semPulsacao++;
        }

        // TRESHOLD PARA ALERTA DE SEM BATIMENTOS
        if (semPulsacao >= 7)
        {
            sendMessage("ALERTA! Sem medições de bpm e SpO2.");
            semPulsacao = 0;
        }
    }
}
```

*Figura 22 - Código loop MAX30100*

Ao iniciar o código *loop*, o chip é actualizado e caso o tempo contado a partir desse momento menos a definição da última medição seja maior que o tempo definido de actualização, ele efectua as leituras.

Caso a leitura da pulsação retorne um valor maior que 0, queremos aumentar o índice do *arraybpm* e colocar a leitura obtida no índice actual. Ao mesmo tempo, temos de reduzir 1 casa ao *arraypulsação*.

De seguida, iniciamos 2 variáveis locais que vão indicar quantos índices contamos (*count*) e qual a soma deles(*sum*). Para isso, esperamos que o *arraybpm* seja maior que o definido, ou seja -1. Adicionamos à soma o valor do *arraybpm* acabado de inserir, e incrementamos o contador.

De seguida, calculamos a média e neste passo, descartamos caso o valor da pulsação seja 0 e imprimimos os valores em serial, novamente por uma questão de visualização. No código final, este passo não será visível.

De seguida, caso a pulsação seja superior a 0, o contador *semPulsação* será mantido a 0. Apenas quando não é detectado um batimento o contador é incrementado por cada segundo que passa. Caso o contador chegue ao *threshold*, é activado o *callmebot*, e o contador é novamente 0, podendo incrementar para enviar nova mensagem caso seja necessário.

```

// INDICA CONEXÃO
Serial.print("A conectar a ");
Serial.println(host);

// ESTABELECEER CONEXÃO COM WIFICLIENT
const int httpPort = 8080;
if (!client.connect(host, httpPort))
{
    Serial.println("conexão falhada");
    return;
}

// CONSTRUÇÃO DA STRING DE ENVIO
String url = "/connectmax?";
url += "bpm=" + String(bpm) + "&";
url += "spo2=" + String(spo2) + "&";
url += "mediabpm=" + String(mediabpm);

// GET REQUEST
Serial.println("A enviar pedido HTTP...");
client.print(String("GET ") + url + "HTTP/1.1\r\n" +
    "Host: " + host + "\r\n" +
    "Connection: close\r\n\r\n");

// THRESHOLD TIMEOUT && ESTABELECEER RERUN
unsigned long timeout = millis();
while (client.available() == 0)
{
    if (millis() - timeout > 1000)
    {
        Serial.println(">>> Client Timeout !");
        client.stop();
        return;
    }
}

// DEVOLVER O QUE O SERVER INDICA
while(client.available())
{
    String line = client.readStringUntil('\r');
    Serial.print(line);
}

// FECHAR CONEXÃO
Serial.println();
Serial.println("Conexão fechada");

// UPDATE TEMPO ULTIMA MEDIÇÃO
ultimaMedicao = millis();
}
}

```

Figura 23 - Código loop MAX30100 (2)

O restante código é uma cópia da ligação ao *Php* do MPU6050, com a alteração da *string* de conexão e do ficheiro que é carregado.

### 3.2.7. PHP MPU6050 & MAX30100

O código *PhP* faz a ligação entre os dados lidos do ESP32 e a nossa base de dados. O código é composto por um conjunto de variáveis que definem o *host*, o nome da DB, o *username* e a *password*. A ligação é estabelecida e o ficheiro recebe os valores para as variáveis que depois liga aos campos da DB. De seguida é preparada um *query*, em que é inserida na tabela que se pretende os valores.

Por último, esta *query* é executada e no *catch* do código retornamos o sucesso ou insucesso na ligação. Os dois códigos *PhP* são idênticos, tendo como únicas diferenças a tabela a que se ligam e os nomes das variáveis não só da DB, mas do ESP32 também.

```
<?php

$host = "192.168.1.72:3306";
$dbname = "esp32";
$username = "esp32max";
$password = "";

try
{
    // CRIAÇÃO DE UM PDO COM AS CREDENCIAIS DA DB
    $dbh = new PDO("mysql:host=$host;dbname=$dbname", $username,
$password);
    // PDO ERRO HANDLE
    $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // IR BUSCAR DADOS DE HTTP + GET REQUEST ESP32
    $bpm = isset($_GET['bpm']) ? $_GET['bpm'] : 0.069;
    $spo2 = isset($_GET['spo2']) ? $_GET['spo2'] : 0.069;
    $mediabpm = isset($_GET['mediabpm']) ? $_GET['mediabpm'] : 0.069;

    // INSERIR DADOS
    $stmt = $dbh->prepare("INSERT INTO max30100 (bpm, spo2, mediabpm)
VALUES (:bpm, :spo2, :mediabpm)");
    $stmt->bindParam(':bpm', $bpm);
    $stmt->bindParam(':spo2', $spo2);
    $stmt->bindParam(':mediabpm', $mediabpm);
    $stmt->execute();

    // MENSAGEM SUCESSO
    echo "Dados inseridos com sucesso";
}
catch(PDOException $e)
{
    // MENSAGEM INSUCESSO
    echo "Conexão falhou: " . $e->getMessage();
}
?>
```

Figura 24 - Código PHP MAX30100

```

<?php

$host = "192.168.1.72:3306";
$dbname = "esp32";
$username = "esp32mpu";
$password = "";

try
{
    // CRIAÇÃO DE UM PDO COM AS CREDENCIAIS DA DB
    $dbh = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
    // PDO ERRO HANDLE
    $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // IR BUSCAR DADOS DE HTTP + GET REQUEST ESP32
    $acceleration_x = isset($_GET['acceleration_x']) ? $_GET['acceleration_x'] : 0.2;
    $acceleration_y = isset($_GET['acceleration_y']) ? $_GET['acceleration_y'] : 0.2;
    $acceleration_z = isset($_GET['acceleration_z']) ? $_GET['acceleration_z'] : 0.2;
    $gyro_x = isset($_GET['gyro_x']) ? $_GET['gyro_x'] : 0.069;
    $gyro_y = isset($_GET['gyro_y']) ? $_GET['gyro_y'] : 0.069;
    $gyro_z = isset($_GET['gyro_z']) ? $_GET['gyro_z'] : 0.069;
    $estado = isset($_GET['estado']) ? $_GET['estado'] : 0.069;

    // INSERIR DADOS
    $stmt = $dbh->prepare("INSERT INTO mpu6050 (acceleration_x, acceleration_y, acceleration_z, gyro_x, gyro_y, gyro_z, estado) VALUES (:acceleration_x, :acceleration_y, :acceleration_z, :gyro_x, :gyro_y, :gyro_z, :estado)");
    $stmt->bindParam(':acceleration_x', $acceleration_x);
    $stmt->bindParam(':acceleration_y', $acceleration_y);
    $stmt->bindParam(':acceleration_z', $acceleration_z);
    $stmt->bindParam(':gyro_x', $gyro_x);
    $stmt->bindParam(':gyro_y', $gyro_y);
    $stmt->bindParam(':gyro_z', $gyro_z);
    $stmt->bindParam(':estado', $estado);
    $stmt->execute();

    // MENSAGEM SUCESSO
    echo "Dados inseridos com sucesso";
}
catch(PDOException $e)
{
    // MENSAGEM INSUCESSO
    echo "Conexão falhou: " . $e->getMessage();
}
?>

```

Figura 25 - Código PHP MPU6050

## 4 - MS PROJECT PRO

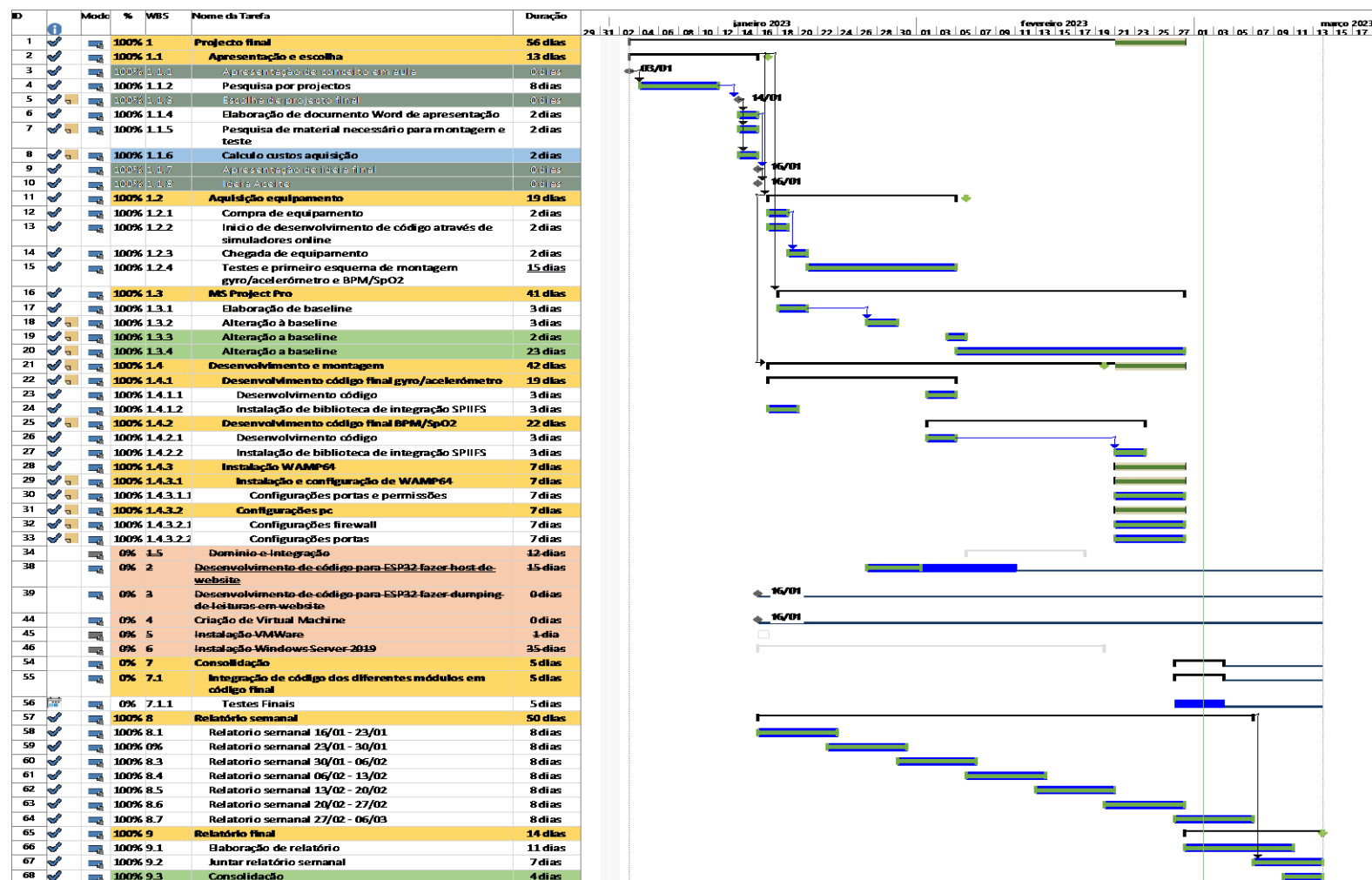


Figura 26 - MS Project Pro

Durante a realização deste trabalho, foi também criado e actualizado semanalmente o ficheiro que acompanha o desenvolvimento e evolução do projecto.

O uso desta ferramenta foi bastante útil porque permitiu seguir as diferenças que foram implementadas ao longo do último mês no projecto, o que em última análise, permitiu facilitar a elaboração deste relatório, uma vez que, conjuntamente com os relatórios semanais (em anexo), deu acesso a uma cronologia do pensamento e das alterações ao projecto final. Foram também realizadas reuniões semanais com o engenheiro Fernando Barros de forma a dar actualizações sobre o estado do projecto e permitir haver troca de ideias e esclarecimentos para assegurar que me encontrava a cumprir, dentro dos prazos estabelecidos, o projecto.

Por último, mencionar que ao utilizar todas estas ferramentas mais as reuniões, tivemos uma simulação de um projecto real numa empresa do mercado de trabalho. Considero que a junção de todas as metodologias foi essencial para a realização com sucesso deste trabalho, uma vez que é importante guardar e catalogar todas as alterações que são realizadas nestes projectos.

Ao longo dos relatórios em anexo é possível ver uma melhoria clara não só no Project Pro, mas na forma como é dado uma explicação para os eventos da semana(s) em questão.



## **5 - RESULTADOS**

Neste trabalho, consegui demonstrar que um microcontrolador é também passível de ser utilizado como um dispositivo médico. Ao criarmos uma DB com informações de um user temos uma base de informação com a qual podemos trabalhar com o objectivo de otimizar ainda mais o nosso programa, e poder providenciar ao utilizador maior conhecimento sobre o seu estado de saúde.

Os resultados obtidos foram assim satisfatórios e de acordo com o que me propôs fazer no início do ano. Considero que os objectivos foram cumpridos, dado as limitações a nível de equipamento e o tempo disponível.

## 6 - CONCLUSÕES

Com este projecto concluimos que não só é possível criar um dispositivo médico a partir de um microcontrolador IoT, mas é necessário face à necessidade de mão de obra que irá existir nos próximos anos em Portugal e no mundo. Aliviar a pressão sobre os sistemas de saúde tem de ser uma prioridade e não existem dúvidas que cada vez mais vão aparecer métodos para tal acontecer.

Os objectivos que me propus ao início foram cumpridos e não acho que haveria muito mais tempo para introduzir complexidade no projecto. No entanto, é importante referir que apenas depois de começar a trabalhar nesta ideia tive noção do mercado e da magnitude do problema que estas soluções vão ter de responder nas próximas décadas.

Estamos a passar por um envelhecimento global, não há nenhum país que seja excepção. Não existe mão de obra qualificada suficiente para responder a todas as necessidades que vão aparecer no mercado. Estas soluções vão ser peça fundamental para o bom funcionamento de qualquer sistema de saúde nos próximos anos e não dar a devida importância a elas pode significar a perda de qualidade de vida para uma grande parte da população em todo o mundo.

A possibilidade de trabalhar os dados dos *users* individualmente é muito importante porque podemos prevenir e antecipar situações que podem comprometer as capacidades do utente, levando a perda de qualidade de vida.

Considero que ao elaborar este relatório, dei uma melhor perspectiva sobre a necessidade destas soluções e desmontei o *proof of concept* que me propus.

## 7 - ANEXOS

### 7.1. RELATÓRIO SEMANAL 16/01 – 27/01

#### MS Project Pro



Figura 27- MS Project Pro 16/01 - 27/01

#### Tarefas antecipadas:

Tarefas
Aquisição de equipamentos
Primeiros Testes
Início de desenvolvimento de código através de simuladores online

Devido à rapidez com que chegaram as peças necessárias, foi já possível fazer a montagem do sistema de giroscópio/acelerómetro e do sistema de medição de pulsação/SpO2x individualmente. Foi possível também fazer os primeiros testes devido a código que adquiri em sites de montagens.

Ao longo da semana fui-me apercebendo que o maior desafio está na interligação de todos os sensores e principalmente a exportação dos dados lidos por os diversos sensores para uma DB.

Irei realizar uma exportação dos dados lidos para um website e vai haver um programa que vai permitir a leitura dos dados e sua importação.

Este vai ser o passo mais complexo e demorado do projecto, como tal, vão ser feitas alterações ao MS Project Pro na próxima semana, para reflectir os novos steps.

## 7.2. RELATÓRIO SEMANAL 30/01 – 06/02

## MS Project Pro

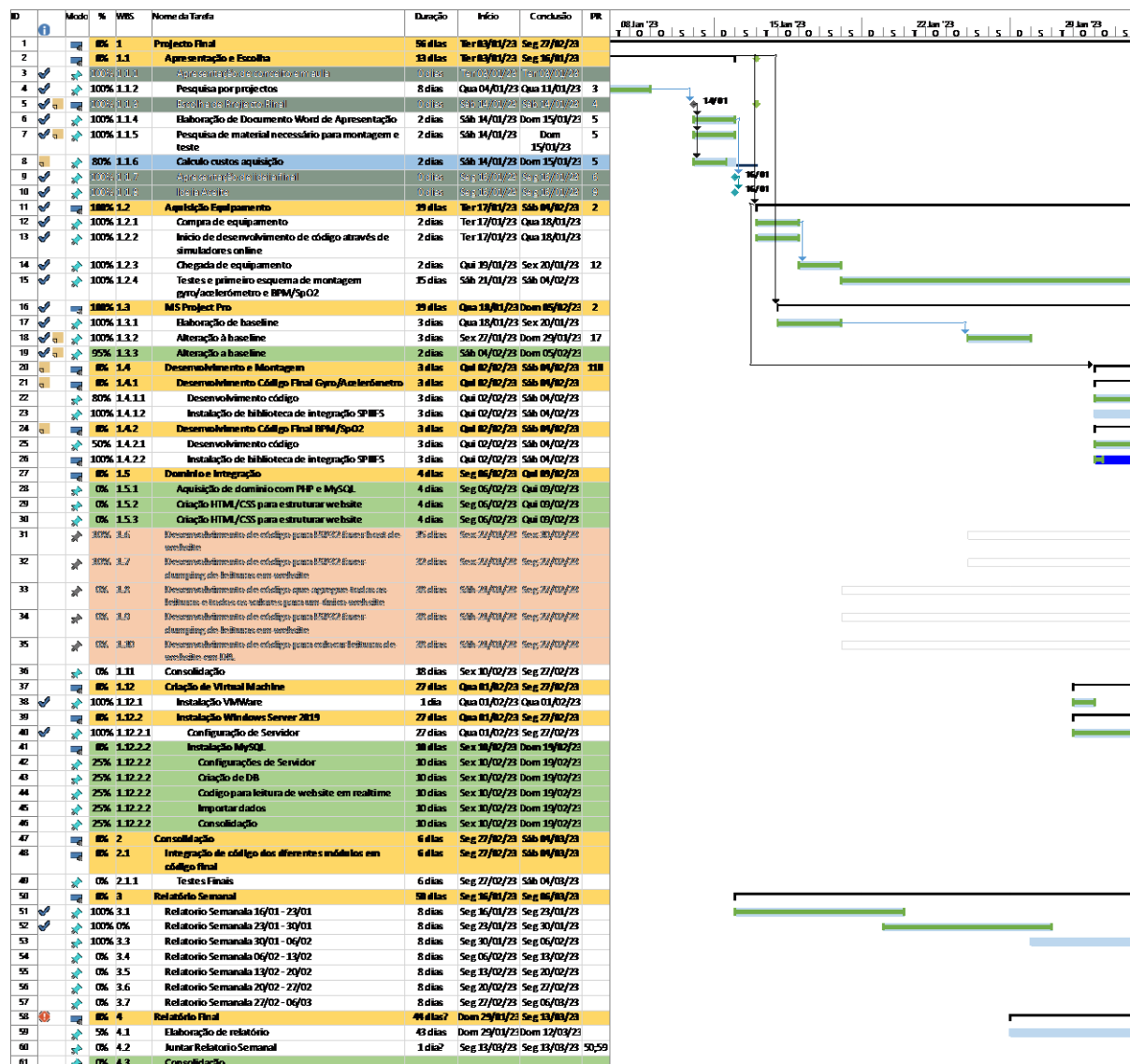


Figura 28 - MS Project Pro 30/01 - 06/02

## MS Project Pro - continuação

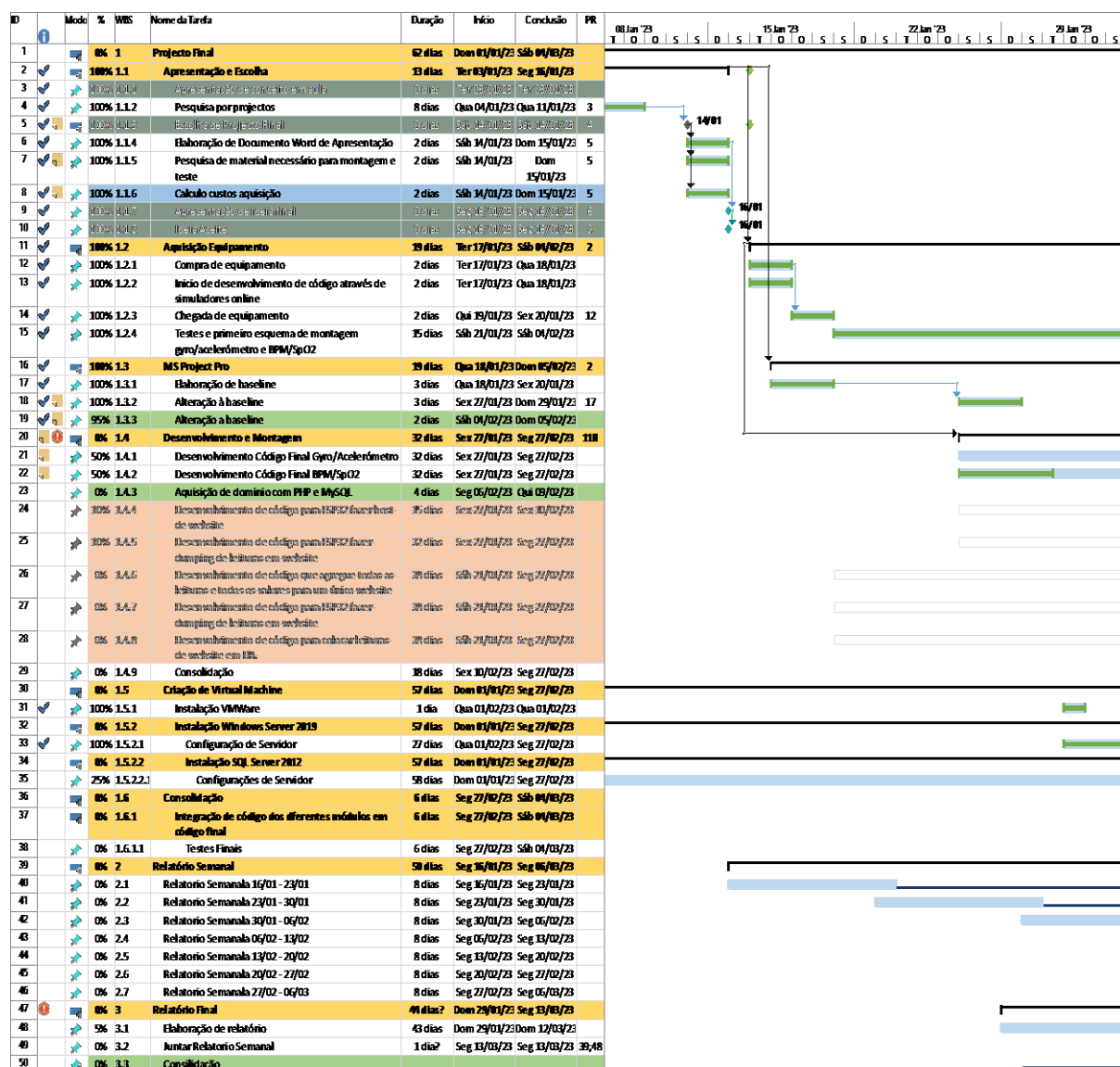


Figura 29 - MS Project Pro 30/01 - 06/02 (2)

Tarefas novas:

Modo	%	WBS	Nome da Tarefa	Duração	Início	Conclusão
Agendada Manualmente	95%	1.3.3	Alteração a baseline	2 dias	Sáb 04/02/23	Dom 05/02/23
Agendada Automaticamente	0%	1.5	Domínio e Integração	4 dias	Seg 06/02/23	Qui 09/02/23
Agendada Manualmente	0%	1.5.1	Aquisição de domínio com PHP e MySQL	4 dias	Seg 06/02/23	Qui 09/02/23
Agendada Manualmente	0%	1.5.2	Criação HTML/CSS para estruturar website	4 dias	Seg 06/02/23	Qui 09/02/23
Agendada Manualmente	0%	4.3	Consolidação			

## Tarefas removidas:

Modo	%	WBS	Nome da Tarefa	Duração	Início	Conclusão
Agendada Manualmente	10%	1.6	<del>Desenvolvimento de código para ESP32 fazer host de website</del>	15 dias	Sex 27/01/23	Sex 10/02/23
Agendada Manualmente	10%	1.7	<del>Desenvolvimento de código para ESP32 fazer dumping de leituras em website</del>	32 dias	Sex 27/01/23	Seg 27/02/23
Agendada Manualmente	0%	1.8	<del>Desenvolvimento de código que agregue todas as leituras e todos os valores para um único website</del>	38 dias	Sáb 21/01/23	Seg 27/02/23
Agendada Manualmente	0%	1.9	<del>Desenvolvimento de código para ESP32 fazer dumping de leituras em website</del>	38 dias	Sáb 21/01/23	Seg 27/02/23
Agendada Manualmente	0%	1.10	<del>Desenvolvimento de código para colocar leituras de website em DB.</del>	38 dias	Sáb 21/01/23	Seg 27/02/23

## Tarefas alteradas:

Modo	%	WBS	Nome da Tarefa	Duração	Início	Conclusão
Agendada Manualmente	80%	1.1.6	Calculo custos aquisição	2 dias	Sáb 14/01/23	Dom 15/01/23

Ao longo desta última semana, foram necessário realizar algumas alterações ao projecto final de forma a este ser exequível.

Primeiro, foi feito um desenvolvimento praticamente total do código necessário para a leitura dos dados do acelerómetro/giroscópio, o aparelho está neste momento com medições correctas, sendo o único problema a calibração dos cálculos do giroscópio, o que é comum com estes chips. É uma questão de encontrar o valor que compensa o coeficiente de desequilíbrio. É também possível ver estes dados em “real time” em web server gerado por o próprio ESP32, utilizando uma biblioteca própria para o efeito.

No caso do sensor BPM/SpO2, existe um problema que ainda não tenho certezas como resolver. O sensor é alimentado por 2 alimentações combinadas, uma para 1.7V que alimenta o LED, necessário para as leituras, e outra de 3.3V o que alimenta o chip em si. O problema é que o ESP32 apenas fornece 3.3V e o chip não sabe o que alimentar, dando origem a leituras incorrectas. A solução passa por ou adquirir um chip mais moderno, ou arranjar uma fonte de alimentação capaz de fornecer os 5V. Também é possível ver estes resultados em web server, mas o desenvolvimento do código não está avançado, uma vez que não é possível confiar nos dados.

Independentemente da solução, existe outro problema que vai levar a alterações profundas na maneira como vou realizar este trabalho. O ESP32 não é capaz, devido as suas limitações de fornecer dados directamente a uma DB.

Sendo assim, vai ser necessário alugar um domínio com protocolo PHP e ligação MySQL para conseguir colocar os dados no website, para que depois através do protocolo seja possível transferir os dados para a DB.

Não vai ser necessário o desenvolvimento de código para isto, sendo que o website já tem a ferramenta “built in”.

Foram também realizadas alterações aos custos, mas para já não existe valor ao certo final.

Neste momento, estou focado no desenvolvimento do código para depois poder tratar do relatório final. Embora esteja dentro dos prazos, o desenvolvimento do relatório vai ficar para o final. Até lá, tenho estes relatórios semanais que também ajudam a perceber os diferentes estados do trabalho



## 7.3. RELATÓRIO SEMANAL 06/02 – 15/02

## MS Project Pro:

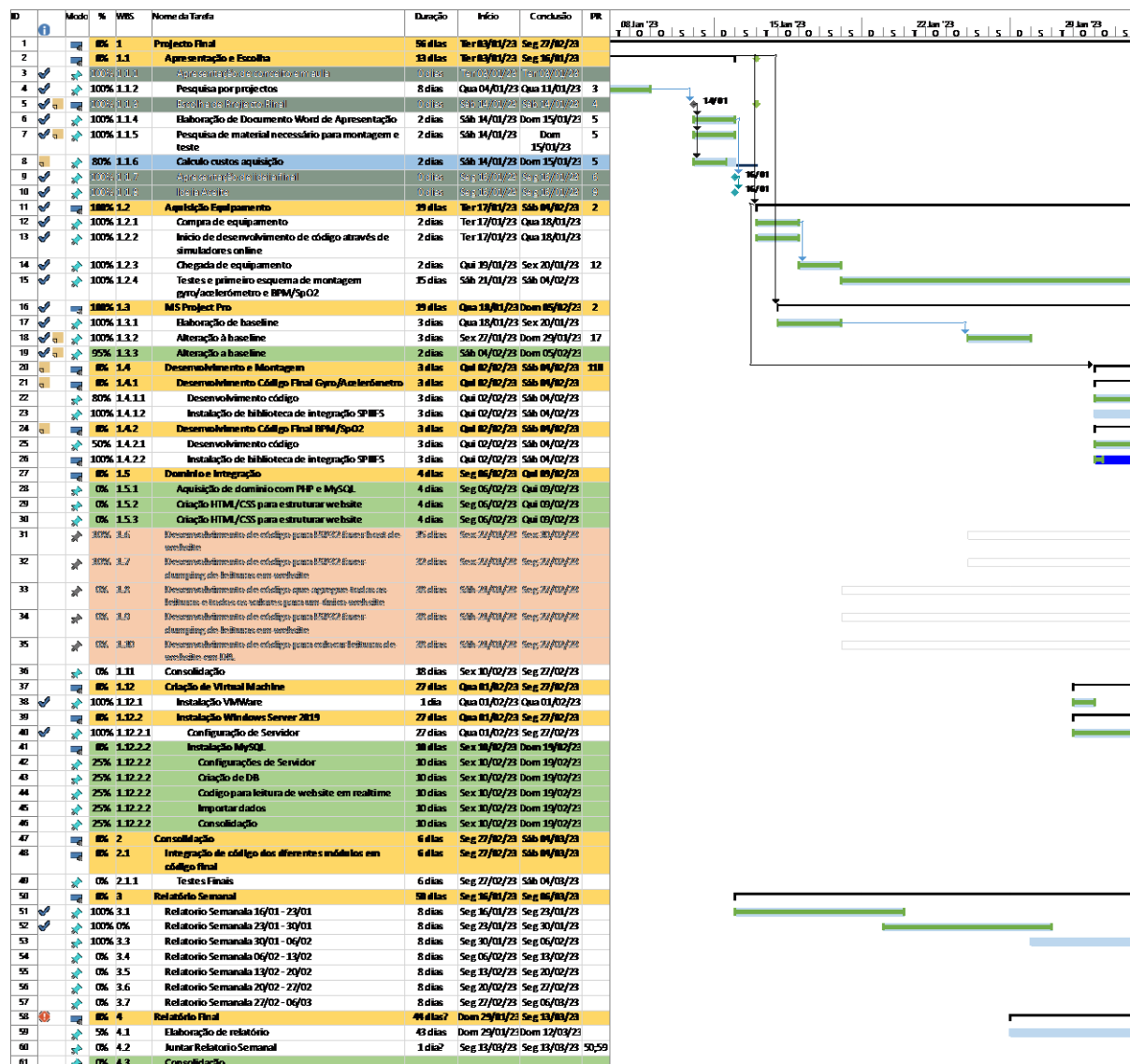


Figura 30 - MS Project Pro 06/02 - 15/02

## MS Project Pro continuação:

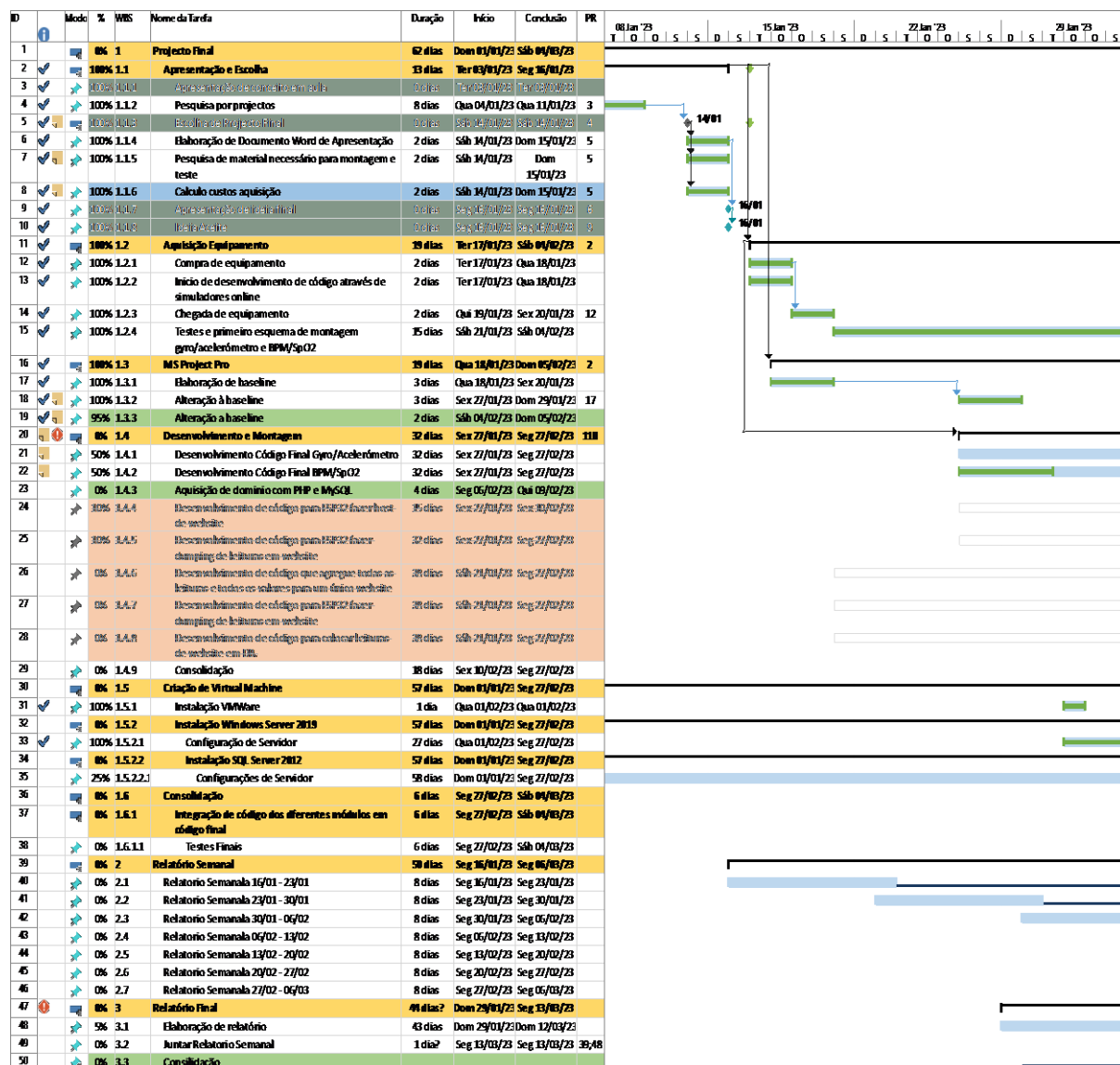


Figura 31 - MS Project Pro 06/02 -15/02 (2)

Esta semana, percebi que devido as limitações do ESP32, apenas consigo fazer a leitura de um dos sensores de cada vez. Isto deve-se ao facto de o chip ter apenas 2 portas com protocolo i2c, o que impossibilita a leitura de vários sensores. As alternativas, são ou comprar um *multiplexer*, ou adquirir um novo ESP32.

Optei por a última, uma vez que não encontrei *multiplexers* à venda com uma data de entrega razoável e que não fizesse com que o projecto se atrase ainda mais.

Para além disto, descobri também que devido à utilização de 2 ESP, o plano de alugar um domínio para fazer upload dos dados é impossível. Uma vez que a DB não consegue ser acedida por 2 aparelhos diferentes e os aparelhos não conseguem colocar na mesma *row* com uma *unique key* os 2 valores. Pode haver também falhas na ligação que comprometem os dados.

Sendo assim, optei por fazer uma DB localmente com os valores de 1 sensor e outra com os valores do outro sensor.

Depois, vai ser necessário programação para interpretar dados de forma a perceber se o *user* esta a andar, de pé, correr ou se teve uma queda. Neste ponto, o projecto começa a ficar bastante atrasado, uma vez que o código todo, mais as verificações vão demorar bastante tempo a serem aperfeiçoados

## 7.4. RELATÓRIO SEMANAL 13/02 – 27/02

## MS Project Pro:

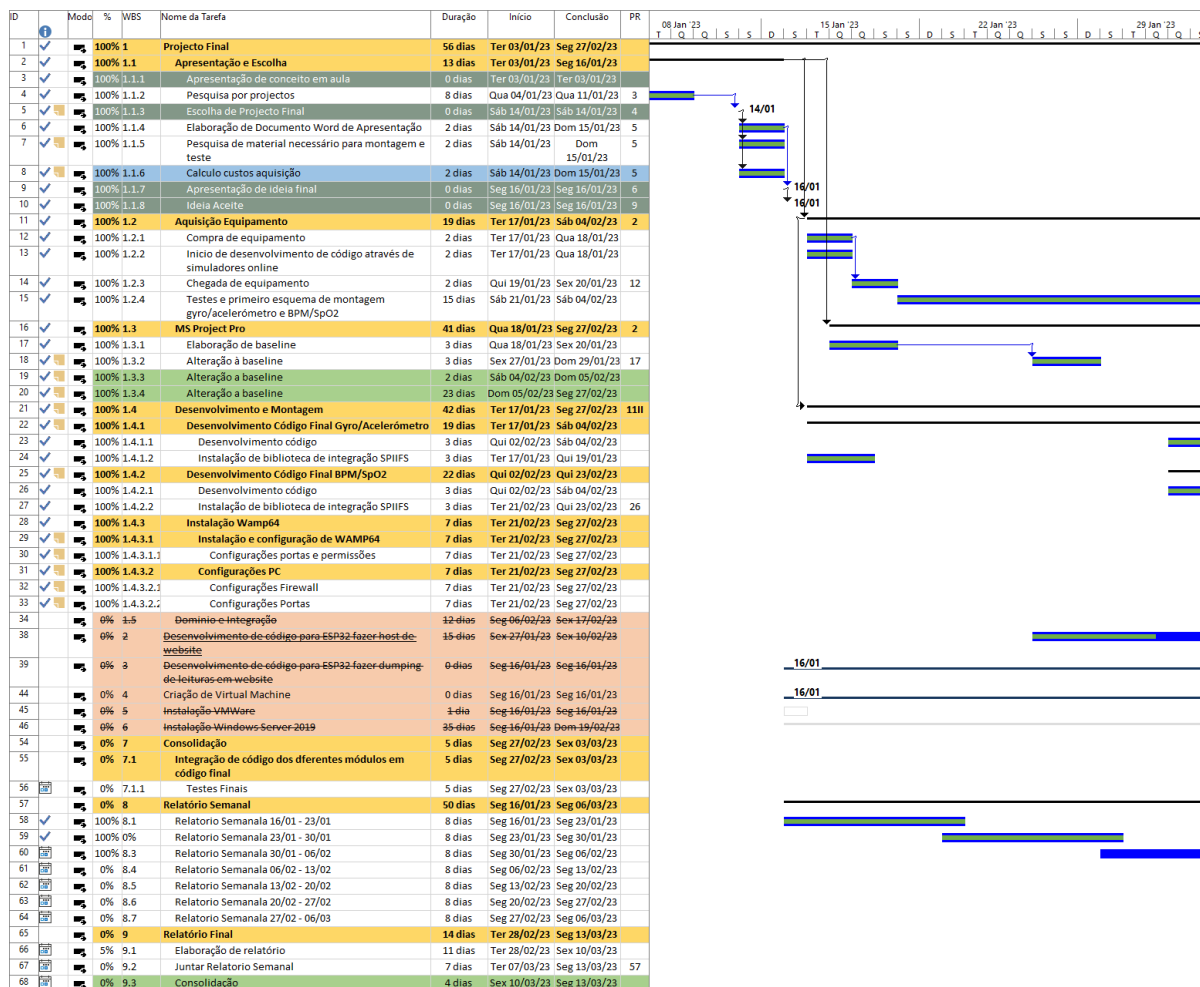


Figura 32- MS Project Pro 13/02 - 27/02

Tarefas novas:

Modo	%	WBS	Nome da Tarefa	Duração	Início	Conclusão
Agendada Automaticamente	100%	1.3.3	Alteração a baseline	2 dias	Sáb 04/02/23	Dom 05/02/23
Agendada Automaticamente	100%	1.3.4	Alteração a baseline	23 dias	Dom 05/02/23	Seg 27/02/23
Agendada Automaticamente	100%	1.4.3	Instalação Wamp64	7 dias	Ter 21/02/23	Seg 27/02/23
Agendada Automaticamente	100%	1.4.3.1	Instalação e configuração de WAMP64	7 dias	Ter 21/02/23	Seg 27/02/23
Agendada Automaticamente	100%	1.4.3.1.1	Configurações portas e permissões	7 dias	Ter 21/02/23	Seg 27/02/23

Tarefas removidas:

Modo	%	WBS	Nome da Tarefa	Duração	Início	Conclusão
<del>Agendada Automaticamente</del>	<del>0%</del>	<del>1.5</del>	<del>Dominio e Integração</del>	<del>12 dias</del>	<del>Seg 06/02/23</del>	<del>Sex 17/02/23</del>
Agendada Manualmente	0%	1.5.1	Aquisição de dominio com PHP e MySQL	4 dias	Seg 06/02/23	Qui 09/02/23
Agendada Manualmente	0%	1.5.2	Criação HTML/CSS para estruturar website	4 dias	Sex 10/02/23	Seg 13/02/23
Agendada Manualmente	0%	1.5.3	Criação HTML/CSS para estruturar website	4 dias	Ter 14/02/23	Sex 17/02/23
<del>Agendada Automaticamente</del>	<del>0%</del>	<del>2</del>	<del>Desenvolvimento de código para ESP32 fazer host de website</del>	<del>15 dias</del>	<del>Sex 27/01/23</del>	<del>Sex 10/02/23</del>
<del>Agendada Automaticamente</del>	<del>0%</del>	<del>3</del>	<del>Desenvolvimento de código para ESP32 fazer dumping de leituras em website</del>	<del>0 dias</del>	<del>Seg 16/01/23</del>	<del>Seg 16/01/23</del>
<del>Agendada Manualmente</del>	<del>0%</del>	<del>3.1</del>	<del>Desenvolvimento de código que agregue todas as leituras e todos os valores para um único website</del>	<del>38 dias</del>	<del>Sáb 21/01/23</del>	<del>Seg 27/02/23</del>
<del>Agendada Manualmente</del>	<del>0%</del>	<del>3.2</del>	<del>Desenvolvimento de código para ESP32 fazer dumping de leituras em website</del>	<del>38 dias</del>	<del>Sáb 21/01/23</del>	<del>Seg 27/02/23</del>
<del>Agendada Manualmente</del>	<del>0%</del>	<del>3.3</del>	<del>Desenvolvimento de código para colocar leituras de website em DB.</del>	<del>38 dias</del>	<del>Sáb 21/01/23</del>	<del>Seg 27/02/23</del>

Modo	%	WBS	Nome da Tarefa	Duração	Início	Conclusão
<del>Agendada Manualmente</del>	<del>0%</del>	<del>3.4</del>	<del>Consolidação</del>	<del>18 dias</del>	<del>Sex 10/02/23</del>	<del>Seg 27/02/23</del>
Agendada Automaticamente	0%	4	Criação de Virtual Machine	0 dias	Seg 16/01/23	Seg 16/01/23
<del>Agendada Automaticamente</del>	<del>0%</del>	<del>5</del>	<del>Instalação VMWare</del>	<del>1 dia</del>	<del>Seg 16/01/23</del>	<del>Seg 16/01/23</del>
<del>Agendada Automaticamente</del>	<del>0%</del>	<del>6</del>	<del>Instalação Windows Server 2019</del>	<del>35 dias</del>	<del>Seg 16/01/23</del>	<del>Dom 19/02/23</del>

Tarefas antecipadas:

Modo	%	WBS	Nome da Tarefa	Duração	Início	Conclusão
<b>Agendada Automaticamente</b>	<b>0%</b>	<b>9</b>	<b>Relatório Final</b>	<b>14 dias</b>	<b>Ter 28/02/23</b>	<b>Seg 13/03/23</b>
Agendada Automaticamente	5%	9.1	Elaboração de relatório	11 dias	Ter 28/02/23	Sex 10/03/23
Agendada Automaticamente	0%	9.2	Juntar Relatório Semanal	7 dias	Ter 07/03/23	Seg 13/03/23
Agendada Automaticamente	0%	9.3	Consolidação	4 dias	Sex 10/03/23	Seg 13/03/23

Ao longo das últimas duas semanas foram feitas alterações cruciais ao projecto.

Foi abandonada a ideia de comprar um domínio em procura por uma solução mais barata e que conseguisse fazer na mesma o *proof of concept* que se pretende.

O desenvolvimento do código está completo, com o ESP32 a enviar mensagens de alerta em caso de quedas/faltas de batimentos por minuto, ao mesmo tempo que guarda os dados numa DB local.

Para isso foi usado o programa WAMP64, que com recurso a MySQL e ao Apache, permitiu emular um website, para poder colocar o ficheiro PHP de forma ao ESP32 poder trabalhar o mesmo, enviando as leituras, sendo depois função do código enviá-las para uma DB a correr num PC local.

Embora não seja o mesmo que o que se pretendia originalmente, é uma simulação aproximada que requer mais complexidade ainda por o simples facto da quantidade de configurações que tem de ser feitos no WAMP. Caso tivesse *webhost* num domínio, era apenas necessário colocar o link para o ESP enviar os dados.

Mencionar também que cheguei à conclusão de que o ESP32 não consegue correr estes 2 chips sem um *multiplexer* de portas I2C. Devido à falta de *multiplexers* e no interesse do tempo, foi feita aquisição de outro ESP32 para fazer leituras separadas entre os BPM/SpO2 e o acelerómetro.

Em relação as tarefas, houve uma grande redução delas, uma vez que já não vai ser necessário a aquisição e configuração do domínio, mas como já tinha mencionado, foram substituídas por umas que mais complexas.

Quanto a tarefas antecipadas, o início da escrita do relatório está agendado para amanhã dia 28/02 o que é uma antecipação de 7 dias face ao projecto inicial.

Apenas mencionar que dado a antecipação, é possível que insira mais complexidade no programa de forma a torná-lo o mais realista possível.



