**RELATÓRIO TÉCNICO**

**TÓPICOS AVANÇADOS 1 – EC020**

Henrique de Paiva Ribeiro

Lucas Reis

Matheus Campos

Samuel Pereira

Prof.º Evandro Luis Brandão Gomes

Santa Rita do Sapucaí, 16 de abril de 2018

1. **Objetivo do projeto**

Este projeto tem como objetivo monitorar algumas informações importantes a respeito dos aviões durante as viagens, tais como a temperatura e a inclinação, para garantir o máximo de segurança durante o voo.

1. **Descrição geral e funções do projeto**

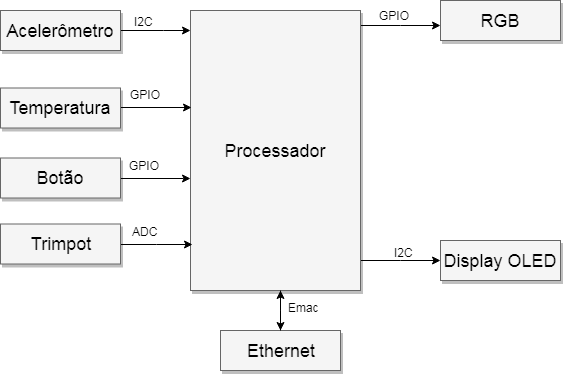
Os aviões são transportes de suma importância na sociedade atual.Visto isso, é necessário garantir o máximo de segurança possível durante os voos, pois qualquer erro pode levar a consequências catastróficas. Diante disso, foi proposto um sistema de monitoramento de informações importantes para o decorrer de uma viagem.

O projeto consiste em coletar informações de inclinação e temperatura do avião. Essas informações são exibidas em um display OLED local e também enviada para um cliente HTTP para que possa ocorrer exibição remota com uma interface amigável.

A temperatura do avião será constantemente monitorada por um sensor de temperatura, que será mostrada no display, como já dito. Além disso, a imagem do céu, ao fundo da figura de um avião, mudará de cor em uma página web que poderá ser acessada remotamente.

O ângulo de inclinação do avião será monitorado pelo acelerômetro, que além de mostrar as informações no display, irá alterar o ângulo da figura do avião na página web. Além disso, um LED RGB indicador poderá ser configurado por meio de um trimpot para mudar de cor de acordo com a inclinação do avião. Essa função do LED poderá ser ativada ou desativada por meio de um botão. Ele fica azul quando a ponta do avião está ligeiramente apontada para cima e vermelho quando a mesma está ligeiramente inclinada para baixo.

1. **Diagrama de blocos do hardware**



1. **Descrição dos sensores**

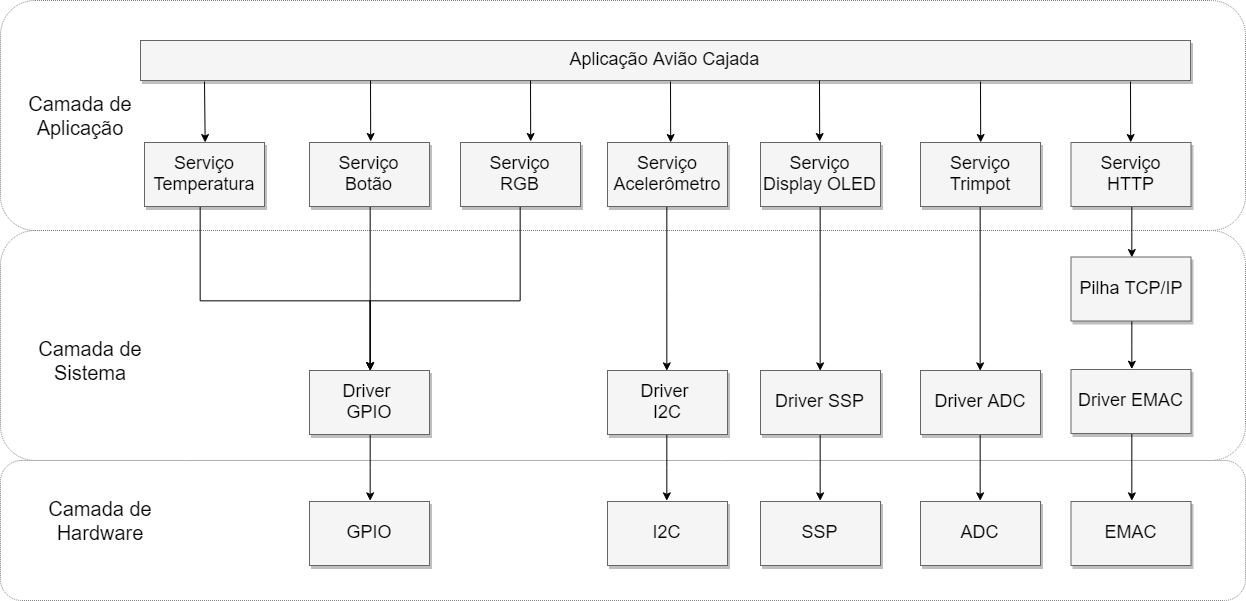
Sensor de temperatura: Possui uma saída PWM, logo sua leitura é feita através de um timer (Systick, nesse caso). Coleta informações a respeito da temperatura e comunica com o controlador por GPIO.

Acelerômetro: Capta informações da inclinação do avião e comunica com o controlador por I2C.

Trimpot: Regula o limite para operação dos leds RGB. Seu valor é lido através do conversor AD, cuja resolução de 14 bits.

Botão: Ativa e desativa o funcionamento do led RGB. Seu valor é acessado através de uma leitura GPIO.

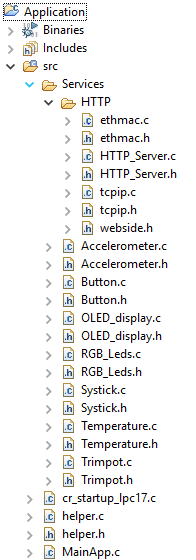
1. **Diagrama de arquitetura do software**

****

1. **Software implementado**

Todo o código do projeto pode ser acessado através do link <https://github.com/samuelpereira7/EC020>.

O projeto está organizado da seguinte maneira no workspace do ambiente de desenvolvimento:



Para o desenvolvimento do firmware, foram utilizadas as bibliotecas CMSISv2p00, Lib\_MCU e Lib\_EaBaseboard. Elas foram incluídas como projetos separados.

Segue abaixo o código da aplicação principal do firmware:

/\*

\* MainApp.c

\*

\* Created on: 01 de abr de 2018

\* Author: samuelpereira

\*/

**#include** "stdlib.h"

**#include** "stdio.h"

**#include** "string.h"

**#include** <Systick.h>

**#include** "Temperature.h"

**#include** "Button.h"

**#include** "Accelerometer.h"

**#include** "Trimpot.h"

**#include** "RGB\_Leds.h"

**#include** "OLED\_display.h"

**#include** "Services/HTTP/tcpip.h"

**#include** <HTTP\_Server.h>

**#include** "helper.h"

**inline** **static** uint8\_t **calc\_threshold**( uint16\_t trimpot\_value );

**int** **main**( **void** )

{

/\* buffer for string operations \*/

uint8\_t buf[30];

int8\_t x = 0;

int8\_t y = 0;

int8\_t z = 0;

int16\_t temp = 0;

uint8\_t button\_status;

uint16\_t trimpot\_value;

uint8\_t RGB\_on = 1;

uint64\_t counter = 0;

uint8\_t threshold;

/\* initializing sensors and actuators \*/

OLED\_display\_init();

Temperature\_init( &getTicks );

Button\_init();

Accelerometer\_init();

Trimpot\_init();

RGB\_Leds\_init();

OLED\_display\_clearScreen();

OLED\_display\_putString( 1, 1, (uint8\_t\*) "+Aviao Cajada+" );

**sprintf**( (**char**\*) buf, " %d.%d.%d.%d", MYIP\_1, MYIP\_2, MYIP\_3, MYIP\_4 );

OLED\_display\_putString( 1, 9, (uint8\_t\*) buf );

OLED\_display\_putString( 1, 25, (uint8\_t\*) "Acc x : " );

OLED\_display\_putString( 1, 33, (uint8\_t\*) "Acc y : " );

OLED\_display\_putString( 1, 41, (uint8\_t\*) "Acc z : " );

OLED\_display\_putString( 1, 49, (uint8\_t\*) "Temp : " );

**while** (1)

{

**if** ((counter % 50000) == 0)

{

HTTP\_Server\_reset();

}

**if** ((counter++ % 4000) == 0)

{

/\* reading sensors \*/

temp = Temperature\_read();

button\_status = Button\_read();

Accelerometer\_read( &x, &y, &z );

trimpot\_value = Trimpot\_read();

/\* calculating the threshold value for y-axis of the accelerometer \*/

threshold = calc\_threshold( trimpot\_value );

/\* RGB leds operations \*/

**if** (button\_status == 0)

{

RGB\_on = !RGB\_on;

}

**if** ((y > threshold) && RGB\_on)

{

RGB\_Leds\_setLeds( RGB\_LEDS\_BLUE );

}

**else** **if** ((y < -threshold) && RGB\_on)

{

RGB\_Leds\_setLeds( RGB\_LEDS\_RED );

}

**else**

{

RGB\_Leds\_setLeds( 0 );

}

/\* displaying info in the oled display \*/

OLED\_display\_fillRect( (1 + 9 \* 6), 17, 80, 24 );

**if** (RGB\_on == 1)

{

OLED\_display\_putString( 1, 17, (uint8\_t\*) "RGB ON " );

}

**else**

{

OLED\_display\_putString( 1, 17, (uint8\_t\*) "RGB OFF" );

}

intToString( x, buf, 10, 10 );

OLED\_display\_fillRect( (1 + 9 \* 6), 25, 80, 32 );

OLED\_display\_putString( (1 + 9 \* 6), 25, buf );

intToString( y, buf, 10, 10 );

OLED\_display\_fillRect( (1 + 9 \* 6), 33, 80, 40 );

OLED\_display\_putString( (1 + 9 \* 6), 33, buf );

intToString( z, buf, 10, 10 );

OLED\_display\_fillRect( (1 + 9 \* 6), 41, 80, 48 );

OLED\_display\_putString( (1 + 9 \* 6), 41, buf );

**sprintf**( (**char**\*) buf, "%d.%dC", temp / 10, temp % 10 );

OLED\_display\_fillRect( (1 + 9 \* 6), 49, 90, 56 );

OLED\_display\_putString( (1 + 9 \* 6), 49, buf );

}

/\* network operations \*/

**if** (!(SocketStatus & SOCK\_ACTIVE))

{

TCPPassiveOpen();

}

DoNetworkStuff();

HTTP\_Server\_process();

}

**return** 0;

}

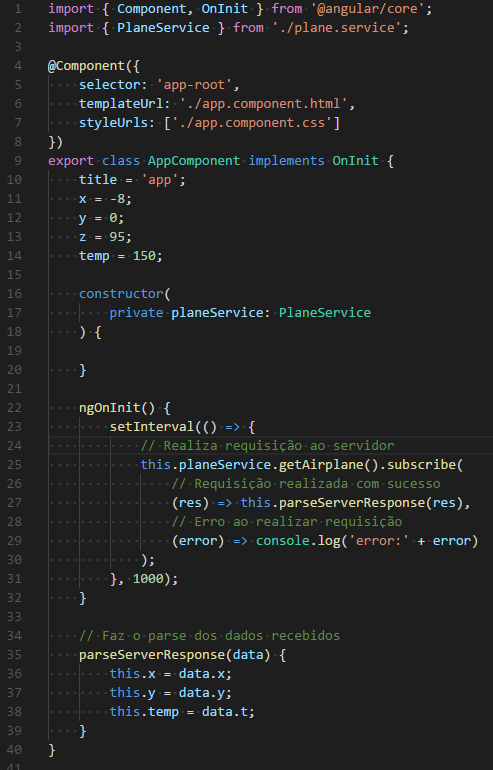
uint8\_t **calc\_threshold**( uint16\_t trimpot\_value )

{

**return** (uint8\_t) (10 + (trimpot\_value / 1000) \* 2);

}

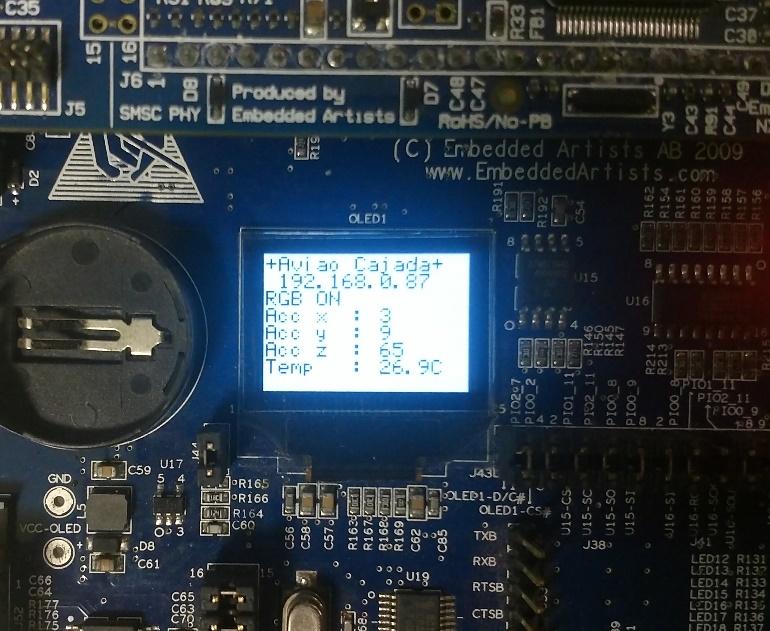
Para criação do cliente, foi utilizado o framework web Angular, segue abaixo código da aplicação principal do front-end, em que são realizadas as requisições para o servidor:



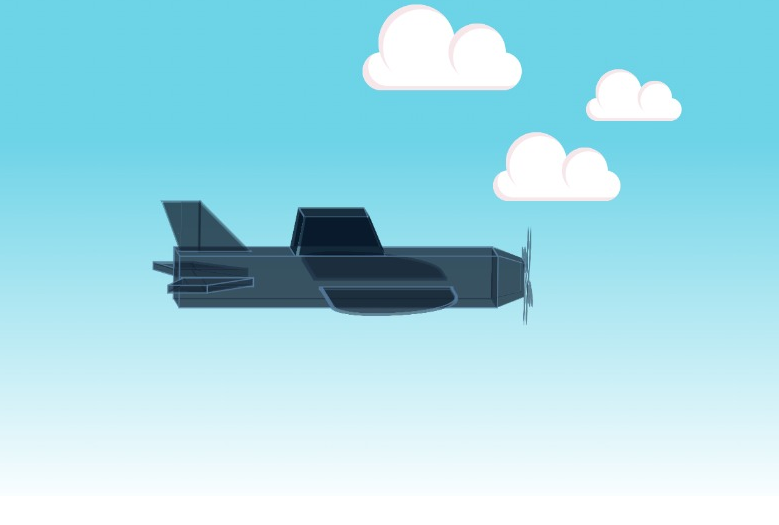
1. **Resultados obtidos**

Os resultados puderam ser analisados na própria baseboard e na aplicação que consome os dados enviados.

Segue abaixo uma fotografia do display OLED. É possível verificar as informações exibidas.

****

Agora, segue um screenshot do website.



1. **Conclusões**

Esse projeto nos proporcionou maior aprendizado sobre arquitetura ARM em geral e também sobre arquiteturas cliente/servidor. Também aprendemos a manipular bibliotecas e como os arquivos de um projeto de firmware se relacionam.

Sobre o escopo do projeto, pode-se concluir que o controle de informações de aviões em tempo real é de extrema importância. Por isso, o projeto é viável não só de ser implementado, como também ser complementado, fazendo análise de ainda mais informações.