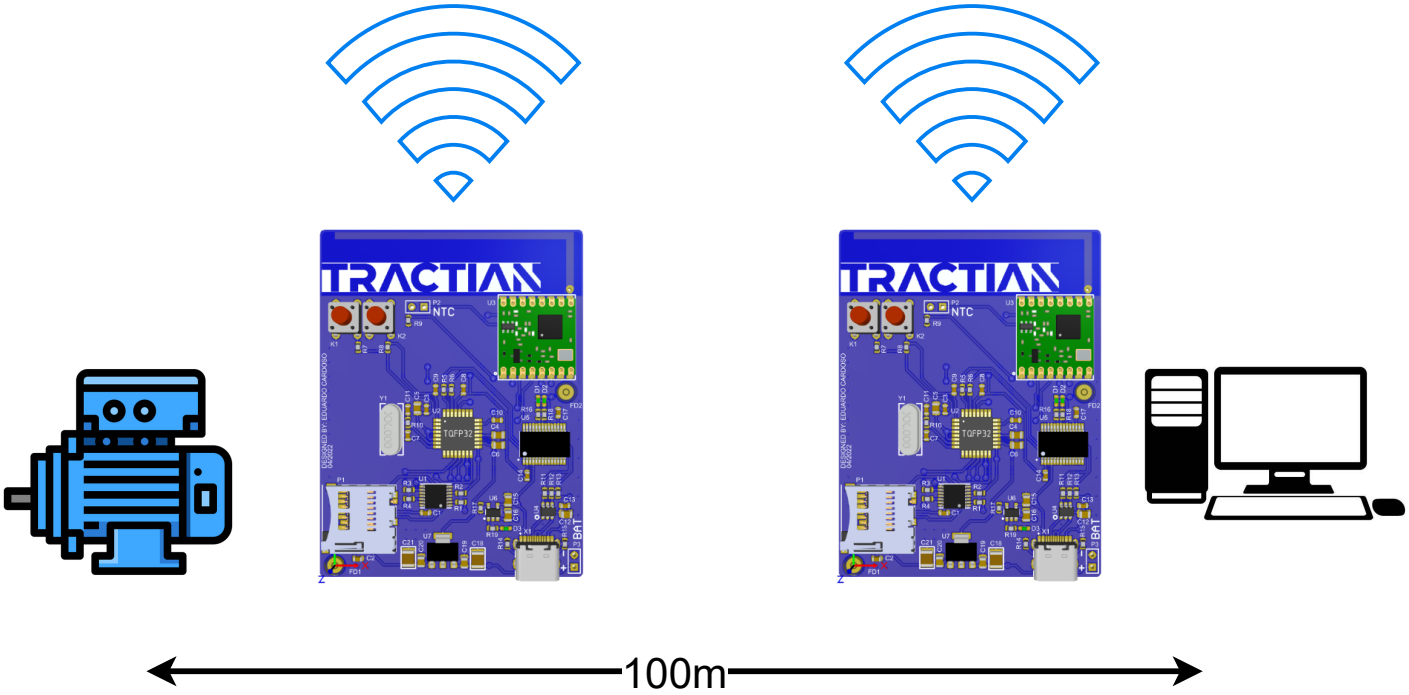
TESTE DE HARDWARE

Objetivo: Desenvolver um sistema de comunicação sem fio completo. Enviar um arquivo de 500kB através de 100m ao ar livre.

Escopo

Para a realização deste teste pensei em uma situação real. Uma indústria que funcione com funcionários somente durante o dia e tenha máquinas em operação durante a noite. Esta empresa necessita monitorar a temperatura de seus motores durante as 12h que fica sem funcionários. Para isso elaborei um hardware escravo que ficara acoplado ao motor junto a um sensor de temperatura. A medição será feita 12h por dia e os dados serão armazenados em uma memória. Após o término das medições e início de um novo turno diurno o arquivo será enviado para um hardware mestre via LoRa. Com isso a empresa pode gerar um relatório de temperatura do motor e fazer o controle de desgastes que podem vir a ser causados por conta de temperaturas em excesso.



EDUARDO CARDOSO

DESENVOLVIMENTO DO HARDWARE

Hardware desenvolvido no software Altium Designer v19.15;

Escravo alimentado por bateria de Lítio 3,7~4,2V;

Mestre alimentado por cabo USB;

Processador ATMEGA328P-AN para ambos;

Sistema de armazenamento de dados via cartão SD;

Sistema de carga de bateria via cabo USB TYPE C;

Módulo Transceptor de RF LoRa RFM96W;

Botão de ativação de início de ciclo;

Sensor de temperatura NTC 10k;

ARQUIVOS DE SAÍDA

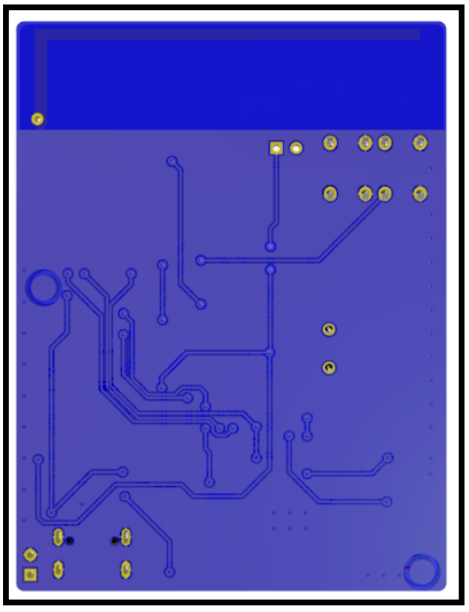
Foram gerados os arquivos de fabricação encontrados dentro em fabricacao.zip;

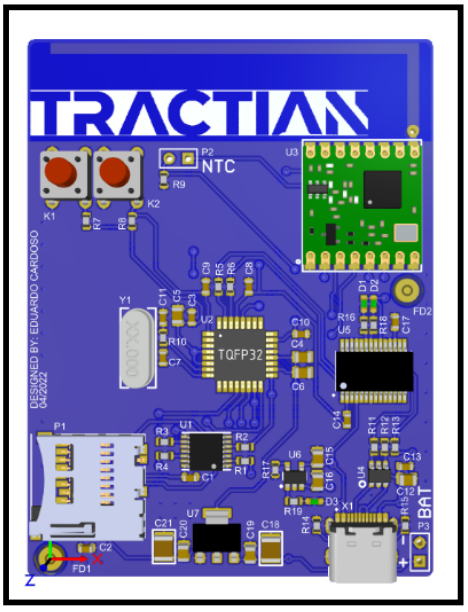
Arquivos Gerber;

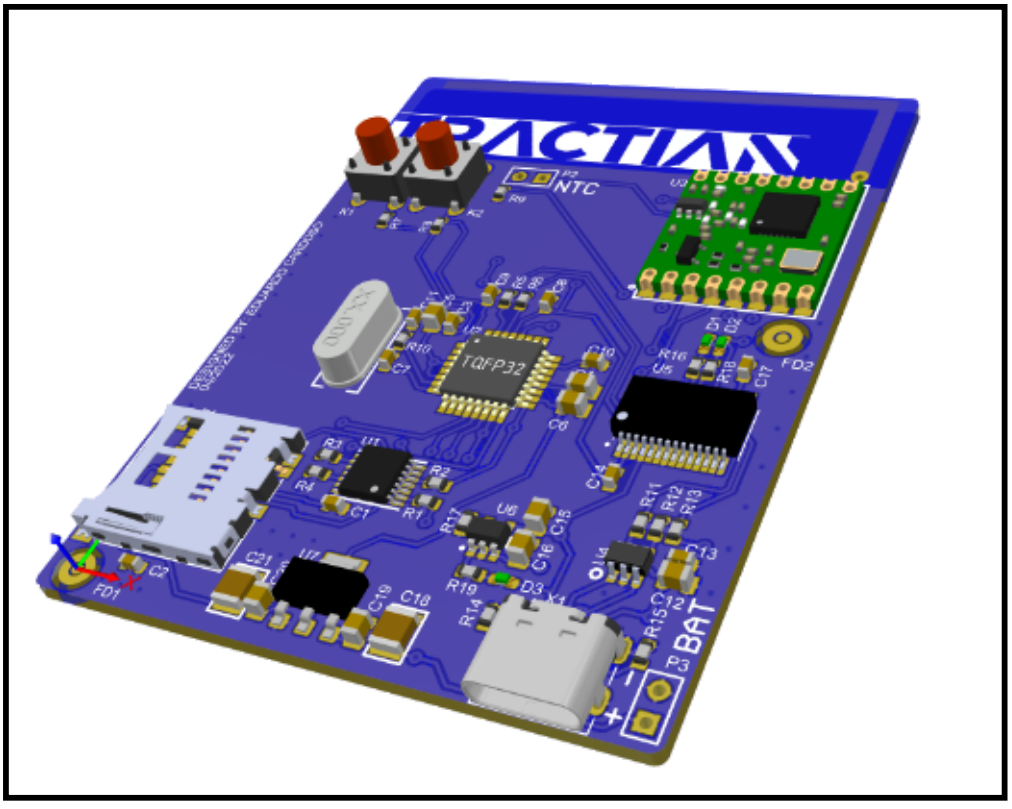
Arquivos NC Drill;

Arquivos PickAndPlace;

Arquivo 3D no formado .step;





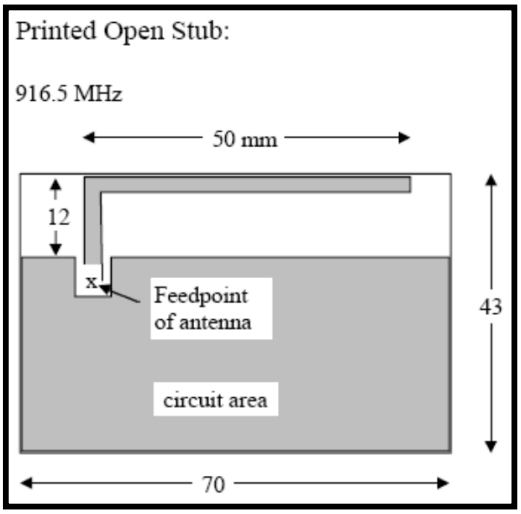


DESENVOLVIMENTO DO HARDWARE

Antena desenvolvida na própria PCB conforme padrão Stub da empresa HOPERF;

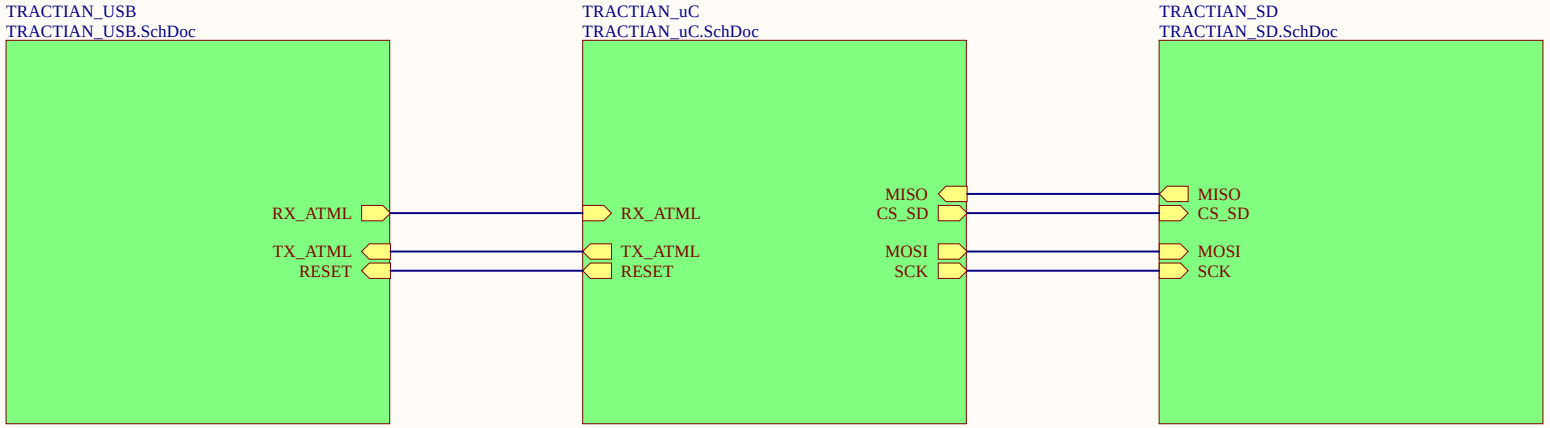
Layout da antena;

Recomendação da HOPERF;

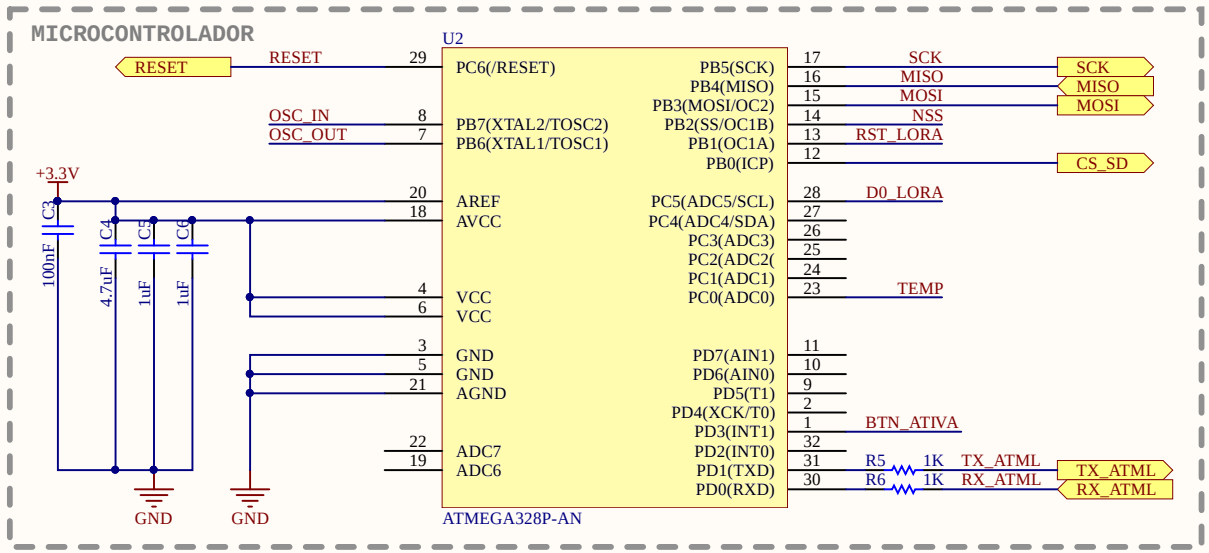


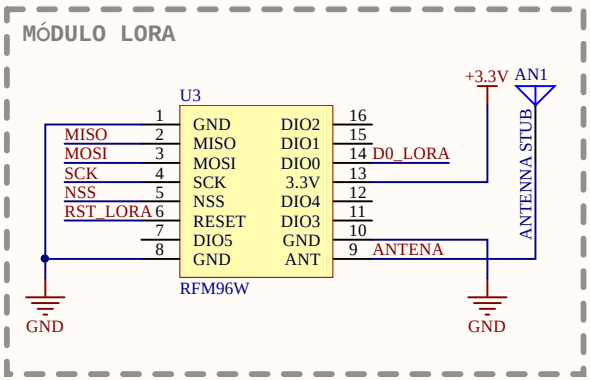
BATERIA

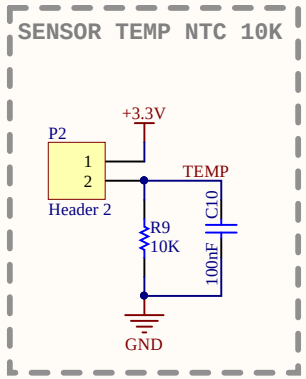
O projeto foi pensado para ser usado com baterias de Lítio (3,7~~4,2V);
Foi implementado um sistema para carga segura das baterias a 500mA, descrito no esquemático do projeto abaixo;

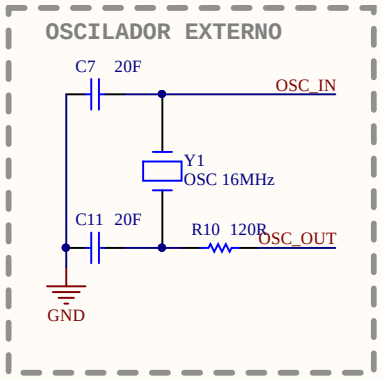


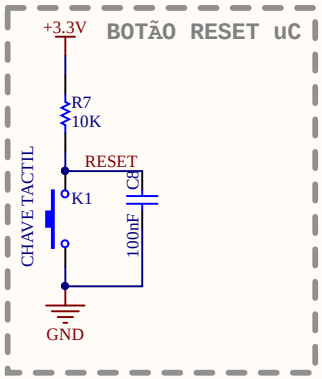
Project: HW TRACTIAN		Sheet: 1 of 4	TRACTIAN
Rev: 01	Date: 05/04/2022		
Author: Eduardo Cardoso			
Approved by: Eduardo Cardoso			

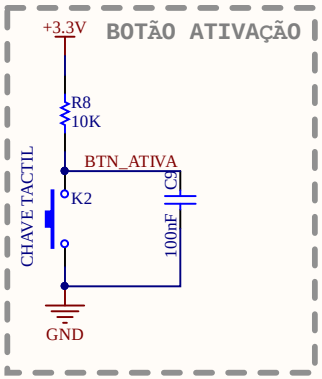










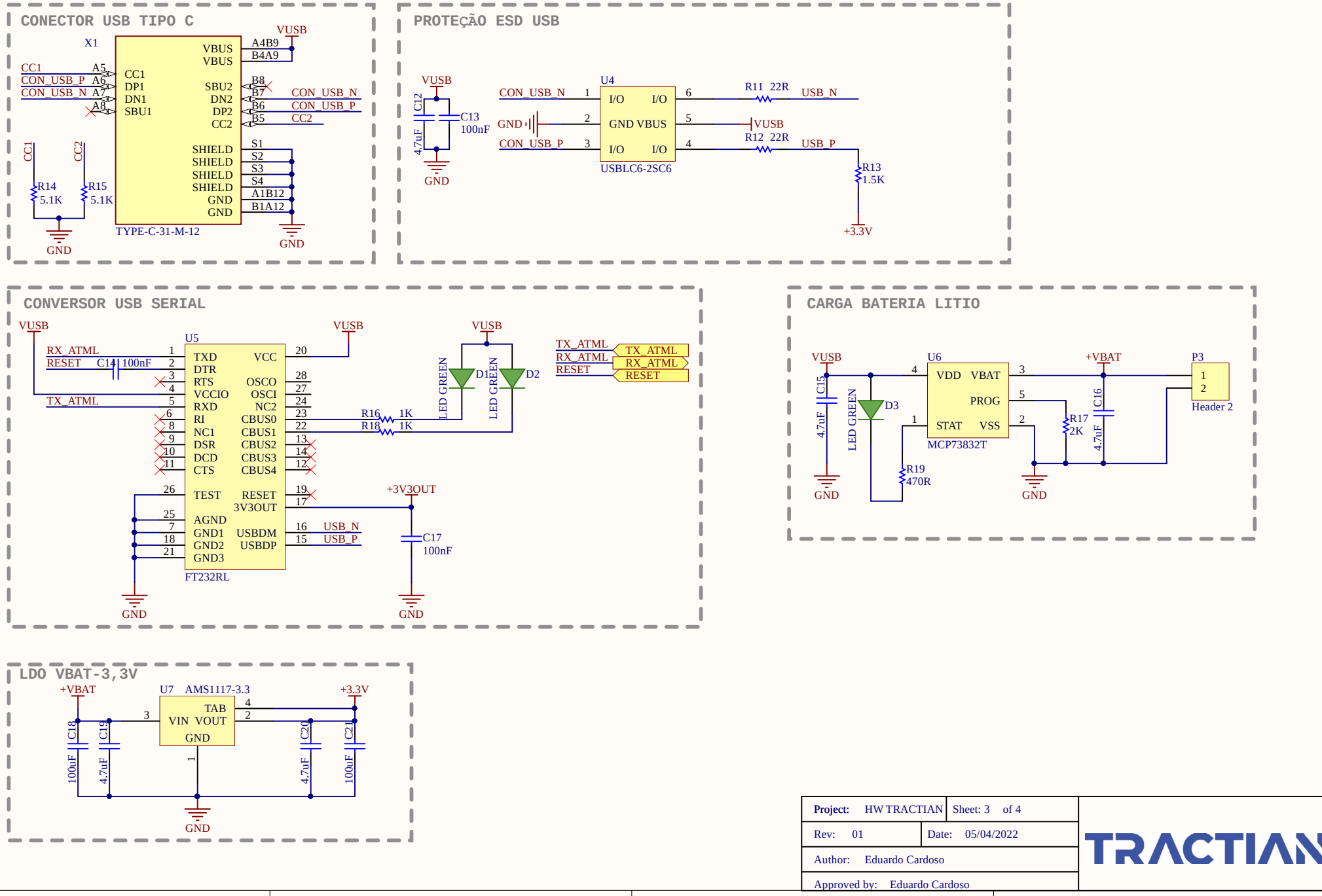


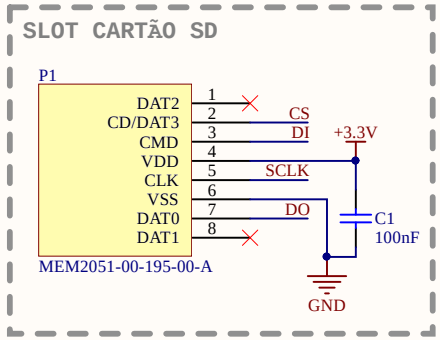
FD1
FIDUCIAL

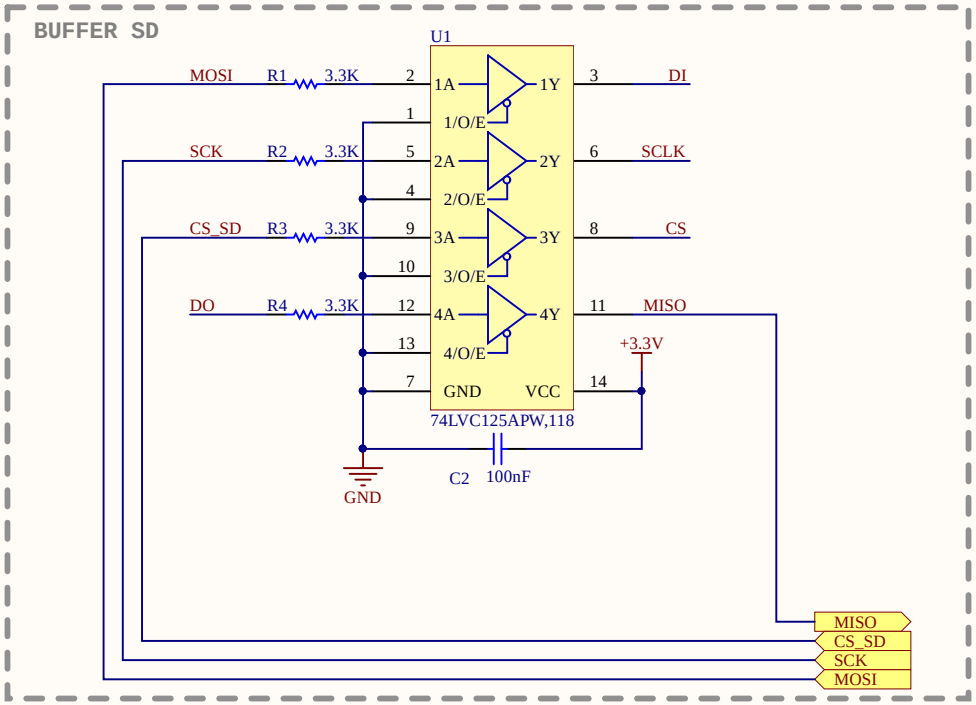
FD2
FIDUCIAL

Project: HW TRACTIAN		Sheet: 2 of 4	
Rev: 01	Date: 05/04/2022		
Author: Eduardo Cardoso			
Approved by: Eduardo Cardoso			

TRACTIAN


CONECTOR USB TIPO C
X1
VUSB
VBUS
VBUS
A4B9
B4A9
CC1
CON_USB_P
CON_USB_N
A5
A6
A7
A8
CC1
DP1
DN1
SBU1
SBU2
DN2
DP2
CC2
B8
B7
B6
B5
CON_USB_N
CON_USB_P
CC2
SHIELD
SHIELD
SHIELD
SHIELD
GND
GND
S1
S2
S3
S4
A1B12
B1A12
CC1
CC2
R14
5.1K
R15
5.1K
GND
TYPE-C-31-M-12
GND
PROTEÇÃO ESD USB
VUSB
C12
4.7uF
C13
100nF
GND
U4
CON_USB_N
GND
CON_USB_P
1
2
3
I/O
GND VBUS
I/O
I/O
6
5
4
I/O
USBLC6-2SC6
R11 22R
USB_N
VUSB
R12 22R
USB_P
R13
1.5K
+3.3V
CONVERSOR USB SERIAL
VUSB
VUSB
VUSB
U5
RX_ATML
RESET
C14
100nF
TX_ATML
1
2
3
4
5
6
8
9
10
11
TXD
DTR
RTS
VCCIO
RXD
RI
NC1
DSR
DCD
CTS
VCC
OSCO
OSCI
NC2
CBUS0
CBUS1
CBUS2
CBUS3
CBUS4
20
28
27
24
23
22
13
14
12
R16
1K
R18
1K
LED GREEN
D1
LED GREEN
D2
TX_ATML
RX_ATML
RESET
TX_ATML
RX_ATML
RESET
26
25
7
18
21
TEST
AGND
GND1
GND2
GND3
RESET
3V3OUT
USBDM
USBDP
19
17
16
15
USB_N
USB_P
+3V3OUT
C17
100nF
FT232RL
GND
GND
CARGA BATERIA LITIO
VUSB
C15
4.7uF
LED GREEN
D3
U6
4
VDD
VBAT
3
PROG
5
STAT
VSS
1
2
MCP73832T
R19
470R
R17
2K
+VBAT
C16
4.7uF
P3
1
2
Header 2
GND
GND
LDO VBAT-3,3V
+VBAT
U7
AMS1117-3.3
3
VIN
TAB
VOUT
GND
4
2
1
+3.3V
C18
100uF
C19
4.7uF
C20
4.7uF
C21
100uF
GND
Project: HW TRACTIAN
Sheet: 3 of 4
Rev: 01
Date: 05/04/2022
Author: Eduardo Cardoso
Approved by: Eduardo Cardoso
TRACTIAN





Project: HW TRACTIAN		Sheet: 4 of 4	
Rev: 01		Date: 05/04/2022	
Author: Eduardo Cardoso			
Approved by: Eduardo Cardoso			

TRACTIAN

DESENVOLVIMENTO DO FIRMWARE

O objetivo do firmware é armazenar um arquivo de **500kB** e enviá-lo por uma rede wireless. Para isso pensei em um arquivo.txt formado basicamente pela temperatura medida de um motor.

A variável de leitura foi declara usando o tipo **uint16_t** (2 bytes);

A leitura é feita a cada 170ms, ou seja, para formarmos um arquivo de 500kB, são feitas 250000 leituras da adc (sensor NTC);

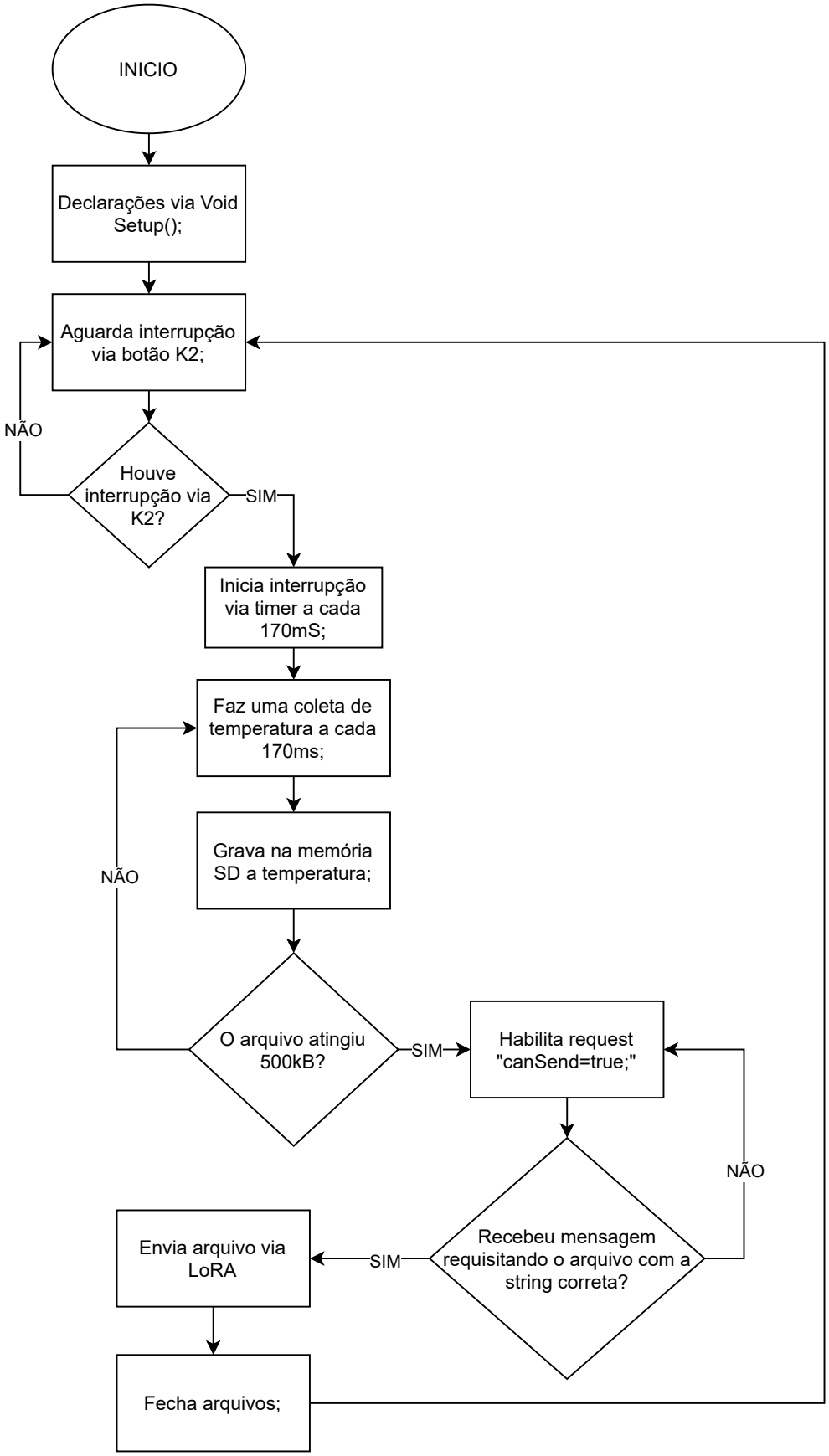
Tamanho: 250000 Leituras * 2 bytes = 500kB;

O tempo total de leitura é: 0,17s * 250000 Leituras = 42500 segundos, ou cerca de 12h (turno noturno);

O arquivo é armazenado em um cartão SD com comunicação SPI com o microcontrolador, isso vale para mestre e escravo;

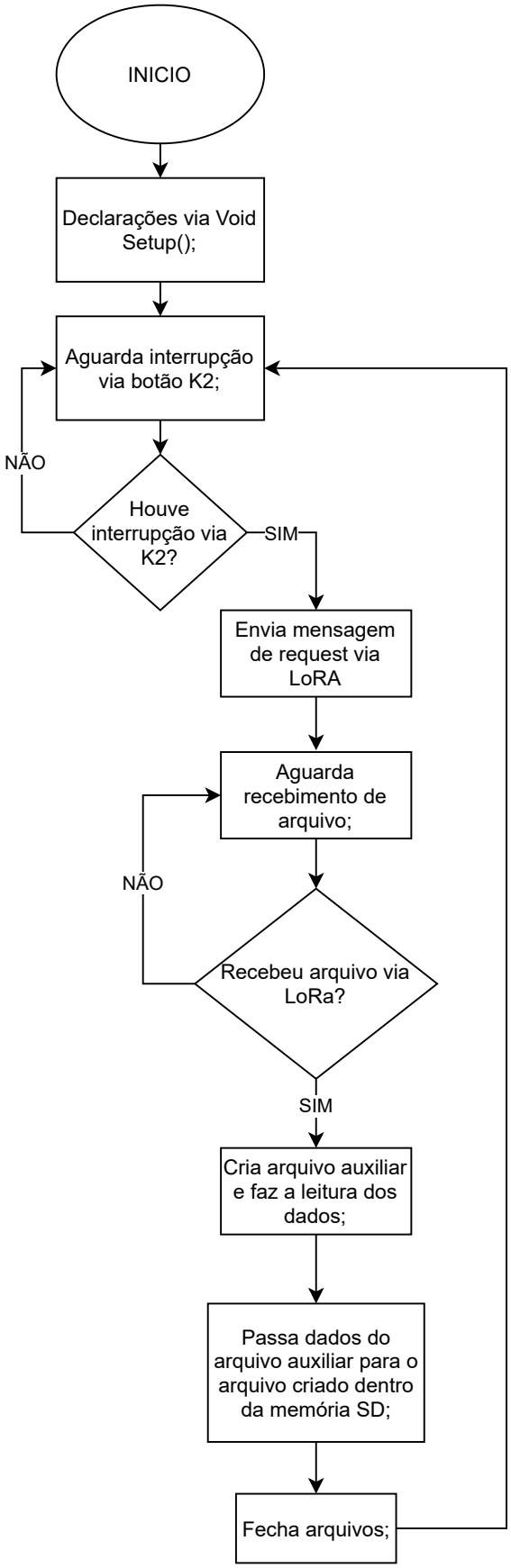
Para o início da leitura é necessário a ação de um funcionário. O mesmo deve apertar o botão K2 do hardware escravo antes de deixar a empresa. Para coleta dos dados pelo hardware mestre um funcionário deve apertar o botão K2 do hardware mestre para gerar um request e fazer a coleta do arquivo;

Fluxograma de funcionamento básico do firmware escravo;



TRACTIAN

Fluxograma de funcionamento básico do firmware mestre;



TRACTIAN

FIRMWARE ESCRAVO

```
/****************************** EDUARDO CARDOSO ***********************************/
/********* FIRMWARE DESENVOLVIDO PARA O TESTE DA EMPRESA TRACTIAN ***************/
/***************************** 05 DE ABRIL DE 2022 ********************************/
/****************************** HARDWARE ESCRAVO **********************************/

/*-----------------------------INCLUDES-------------------------------------------*/
#include <TimerOne.h>
#include <SD.h>
#include <SPI.h>
#include <LoRa.h>
#include "thermistor.h"
/*--------------------------------------------------------------------------------*/

/*-----------------------------DEFINES--------------------------------------------*/
#define btn_ativa              3    //Pino da entrada digital D3
#define adcTemp               A0    //Pino da entrada analogica A0
#define pinSS_SD               8    //Pino SS cartão SD D8
#define csPin                 10    //Pino SS módulo LoRa D10
#define resetPin               9    //Reset do modulo LoRa D9
#define irqPin                A5   //Pino DI0 módulo LoRa A5
/*--------------------------------------------------------------------------------*/
/*-----------------------------VARIAVEIS------------------------------------------*/
File myFile;
uint32_t contRead;
uint16_t Temp;
uint8_t readyFile=0;

byte localAddress = 0xBB;     // ENDEREÇO ESCRAVO LoRa
byte msgCount = 0;            // CONTADOR DE MENSAGENS ENVIADAS
byte destination = 0xFF;      // ENDEREÇO MESTRE LoRa
long lastSendTime = 0;        // TimeStamp DA ULTIMA MENSAGEM ENVIADA
int interval = 5000;          // INTERVALO EM ms NO ENVIO DAS MENSAGENS 5s
uint8_t canSend;
/*--------------------------------------------------------------------------------*/
/*-----------------------------OBJETOS--------------------------------------------*/
THERMISTOR thermistor(adcTemp,        // PINO
                      10000,          // RESISTENCIA NOMINAL A 25 °C
                      3950,           // COEFICIENTE BETA
                      10000);         // VALOR DE RESISTOR SERIE
/*--------------------------------------------------------------------------------*/

void onReceive(int packetSize) //FUNÇÃO DE RECEBIMENTO LoRa
{
  if (packetSize == 0) return;          // SE NENHUMA MENSAGEM FOR RECEBIDA RETORNA ZERO

  // Leu um pacote, vamos decodificar?
  int recipient = LoRa.read();          // ENDEREÇO DE QUEM RECEBE
  byte sender = LoRa.read();            // ENDEREÇO DE QUEM ENVIA
  byte incomingMsgId = LoRa.read();     // MENSAGEM
  byte incomingLength = LoRa.read();    // TAMANHO DA MENSAGEM

  String incoming = "";

  while (LoRa.available())
  {
    incoming += (char)LoRa.read();
  }

  if (incomingLength != incoming.length())
  {
    Serial.println("ERRO!DIVERGENCIA NO TAMANHO DA MENSAGEM");//PRINTA DIVERGÊNCIA
    return;
  }

  if (recipient != localAddress && recipient != 0xFF)
  {
    Serial.println("ERRO!ENDERECO INCORRETO");//PRINTA ERRO DE ENDEREÇO
    return;
  }

  /*PRINTA DETALHES DA MENSAGEM*/
  Serial.println("Recebido do dispositivo: 0x" + String(sender, HEX));
  Serial.println("Enviado para: 0x" + String(recipient, HEX));
  Serial.println("ID da mensagem: " + String(incomingMsgId));
  Serial.println("Tamanho da mensagem: " + String(incomingLength));
  Serial.println("Mensagem: " + incoming);
  Serial.println("RSSI: " + String(LoRa.packetRssi()));
  Serial.println("Snr: " + String(LoRa.packetSnr()));
  Serial.println();

    if(incoming=="SENDME" && readyFile==1){//VERIFICA SE PODE HABILITAR O ENVIO DO ARQUIVO
        canSend= true;
    }else{
        canSend=false;
    }
}
```

```
void sendFile(File _file) // FUNÇÃO DE ENVIO VIA LoRa
{
  LoRa.beginPacket();                   // INICIA PACOTE
  LoRa.write(destination);              // ENDEREÇO DE DESTINO
  LoRa.write(localAddress);             // ENDEREÇO DE QUEM ENVIA
  LoRa.write(msgCount);                 // CONTADOR DE MENSAGEM
  LoRa.write(_file);                    // ARQUIVO
  LoRa.endPacket();                     // FINALIZA PACOTE
  msgCount++;                           // CONTADOR DE MENSAGENS ENVIADAS
}

void readAdc(){
    contRead++;
    if(contRead>=2500000){
        contRead=0;
        Timer1.detachInterrupt(); // DESABILITA INTERRUPÇÃO
        myFile.close();
        readyFile=1;
    }else{
        Temp=thermistor.read();   // LÊ TEMPERATURA
        myFile.println(Temp); // ESCREVE NO ARQUIVO
        readyFile=0;
    }

}
void readTemp(){
    Timer1.initialize(170000); // INICIA TIMER COM PERIODO DE 170ms
    Timer1.attachInterrupt(readAdc); // CHAMA FUNÇÃO
}

void setup() {
   Serial.begin(9600);
   pinMode(irqPin, OUTPUT);
  /*************** SETUP INTERRUPÇÃO **********************************/
  pinMode(btn_ativa,INPUT);
  attachInterrupt(digitalPinToInterrupt(btn_ativa),readTemp,FALLING);
  /********************************************************************/

  /*************** SETUP SD CARD **************************************/
  if (SD.begin()) { // Inicializa o SD Card
       Serial.println("SD Card pronto para uso.");
    }else {
        Serial.println("Falha na inicialização do SD Card.");
    return;
    }
    myFile = SD.open("LOG_TEMP_TRACTIAN.txt", FILE_WRITE); //CRIA ARQUIVO .txt
  /********************************************************************/

  /*************** SETUP MÓDULO LORA **********************************/
   LoRa.setPins(csPin, resetPin, irqPin);
  if (!LoRa.begin(915E6))
  {
    Serial.println(" Erro ao iniciar modulo LoRa. Verifique a coenxao dos seus pinos!");
    while (true);
  }
  /********************************************************************/
}

void loop() {
  onReceive(LoRa.parsePacket());//FUNÇÃO PARA RECEBER O REQUEST DO MESTRE
  if(canSend==true){
     if (millis() - lastSendTime > interval)// INTERVALO ENTRE MENSAGENS
    {
        sendFile(myFile);
        lastSendTime = millis();
    }
  }
}
```

TRACTIAN

FIRMWARE MESTRE

```
/****************************** EDUARDO CARDOSO *********************************/
/********* FIRMWARE DESENVOLVIDO PARA O TESTE DA EMPRESA TRACTIAN **************/
/************************** 05 DE ABRIL DE 2022 *********************************/
/****************************** HARDWARE MESTRE *********************************/

/*----------------------------INCLUDES-----------------------------------------*/
#include <TimerOne.h>
#include <SD.h>
#include <SPI.h>
#include <LoRa.h>
#include "thermistor.h"
/*-----------------------------------------------------------------------------*/

/*----------------------------DEFINES_-----------------------------------------*/
#define btn_ativa              3    //Pino da entrada digital D3
#define adcTemp               A0    //Pino da entrada analogica A0
#define pinSS_SD               8    //Pino SS cartão SD D8
#define csPin                 10    //Pino SS módulo LoRa D10
#define resetPin               9    //Reset do modulo LoRa D9
#define irqPin                A5   //Pino DI0 módulo LoRa A5
/*-----------------------------------------------------------------------------*/
/*----------------------------VARIAVEIS----------------------------------------*/
File myFile;
uint32_t contRead;
uint8_t readyFile=0;

String outgoing;              // ARMAZENA MENSAGEM

byte localAddress = 0xFF;     // ENDEREÇO ESCRAVO LoRa
byte msgCount = 0;            // CONTADOR DE MENSAGENS ENVIADAS
byte destination = 0xBB;      // ENDEREÇO MESTRE LoRa
long lastSendTime = 0;        // TimeStamp DA ULTIMA MENSAGEM ENVIADA
int interval = 5000;          // INTERVALO EM ms NO ENVIO DAS MENSAGENS 5s
uint8_t canReceived;
/*-----------------------------------------------------------------------------*/
 void onReceive(int packetSize)
 {
   if (packetSize == 0) return;          // SE NENHUMA MENSAGEM FOR RECEBIDA RETORNA ZERO

   // Leu um pacote, vamos decodificar?
   int recipient = LoRa.read();          // ENDEREÇO DE QUEM RECEBE
   byte sender = LoRa.read();            // ENDEREÇO DE QUEM ENVIA
   byte incomingMsgId = LoRa.read();     // MENSAGEM

   File receveidFile; // CRIA ARQUIVO AUX
   uint16_t tempFile;

   while(receveidFile.available())//TRANSFERE AUX PARA SD
   {
     tempFile = receveidFile.read();
     myFile.println(tempFile) ;

   }

   if (recipient != localAddress && recipient != 0xFF)
   {
     Serial.println("ERRO!ENDERECO INCORRETO");//PRINTA ERRO DE ENDEREÇO
     return;
   }

   receveidFile.close();
   canReceived = false;

   /*PRINTA DETALHES DA MENSAGEM*/
   Serial.println("Recebido do dispositivo: 0x" + String(sender, HEX));
   Serial.println("Enviado para: 0x" + String(recipient, HEX));
   Serial.println("ID da mensagem: " + String(incomingMsgId));
   Serial.println("RSSI: " + String(LoRa.packetRssi()));
   Serial.println("Snr: " + String(LoRa.packetSnr()));
   Serial.println();
 }
```

```
void sendMessage(String outgoing) // FUNÇÃO DE ENVIO MENSAGEM VIA LoRa
{
  LoRa.beginPacket();                   // Inicia o pacote da mensagem
  LoRa.write(destination);              // Adiciona o endereco de destino
  LoRa.write(localAddress);             // Adiciona o endereco do remetente
  LoRa.write(msgCount);                 // Contador da mensagem
  LoRa.write(outgoing.length());        // Tamanho da mensagem em bytes
  LoRa.print(outgoing);                 // Vetor da mensagem
  LoRa.endPacket();                     // Finaliza o pacote e envia
  msgCount++;                           // Contador do numero de mensagnes enviadas
}


void requestFile(){
    canReceived=true;
}
void setup() {
    Serial.begin(9600);
  /*************** SETUP INTERRUPÇÃO ***********************************/
  pinMode(btn_ativa,INPUT);
  attachInterrupt(digitalPinToInterrupt(btn_ativa),requestFile,FALLING);
  /*********************************************************************/

  /*************** SETUP SD CARD ***************************************/
  if (SD.begin()) { // Inicializa o SD Card
        Serial.println("SD Card pronto para uso."); // Imprime na tela
    }else {
        Serial.println("Falha na inicialização do SD Card.");
    return;
    }
  /*********************************************************************/

  /*************** SETUP MÓDULO LORA ***********************************/
   LoRa.setPins(csPin, resetPin, irqPin);

  if (!LoRa.begin(915E6))
  {
    Serial.println(" Erro ao iniciar modulo LoRa. Verifique a coenxao dos seus pinos!");
    while (true);
  }
  /*********************************************************************/
}

void loop() {

  if(canReceived==true){
    if (millis() - lastSendTime > interval)
  {
    String mensagem = "SENDME";//ENVIA MENSAGEM
    sendMessage(mensagem);
    lastSendTime = millis();
  }
  }
    onReceive(LoRa.parsePacket());//FUNÇÃO PARA RECEBER O ARQUIVO DO ESCRAVO
}
```

TRACTIAN