INSTITUTO FEDERAL
Santa Catarina

ESTUFA HIDROPÔNICA VERTICAL INTELIGENTE

Instituto Federal de Santa Catarina

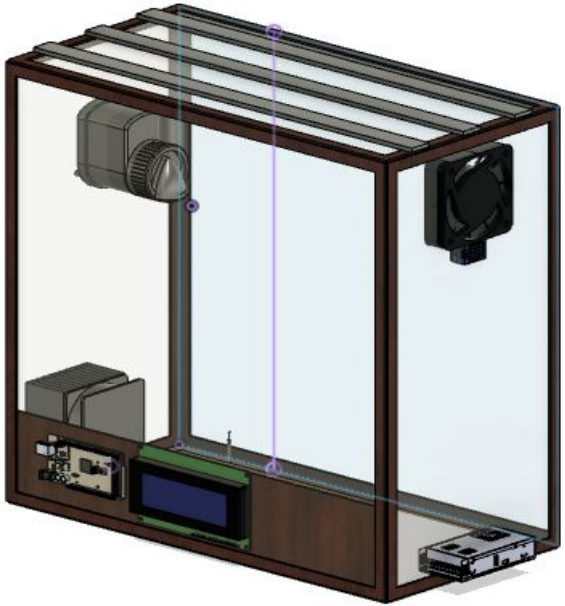
Engenharia Eletrônica
Projeto Integrador 2

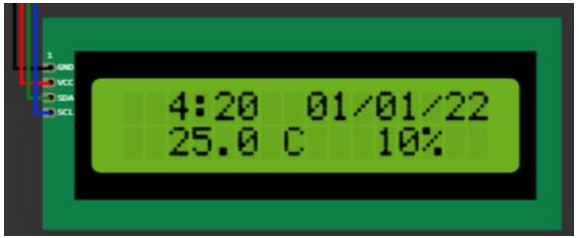
Mateus Salgado Barboza Costa

INSTITUTO FEDERAL
Santa Catarina

IDEIA INICIAL

ESTUFA HIDROPÔNICA VERTICAL INTELIGENTE - Projeto Integrador 1

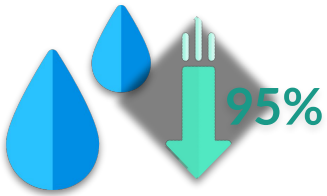


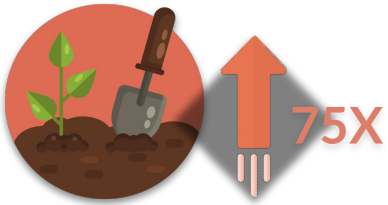


INSTITUTO FEDERAL
Santa Catarina

OBJETIVO DO PROJETO

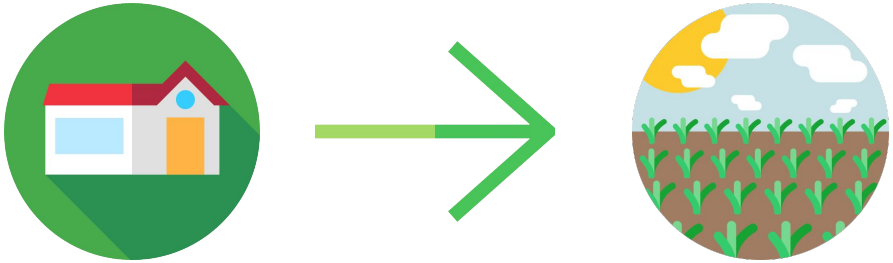
Recursos Ambientais e Controle de Ambiente





PÚBLICO ALVO

- Para residências, mas a tecnologia pode ser expandida para uma plantação maior.

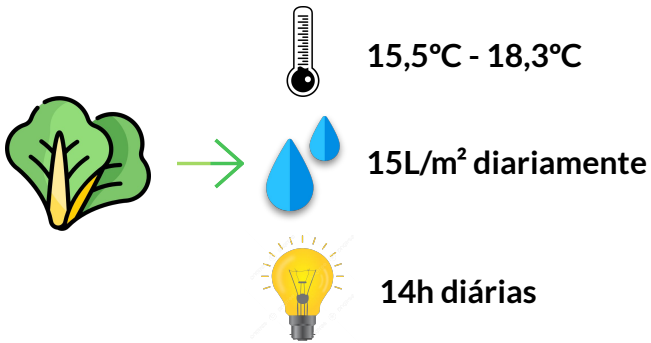


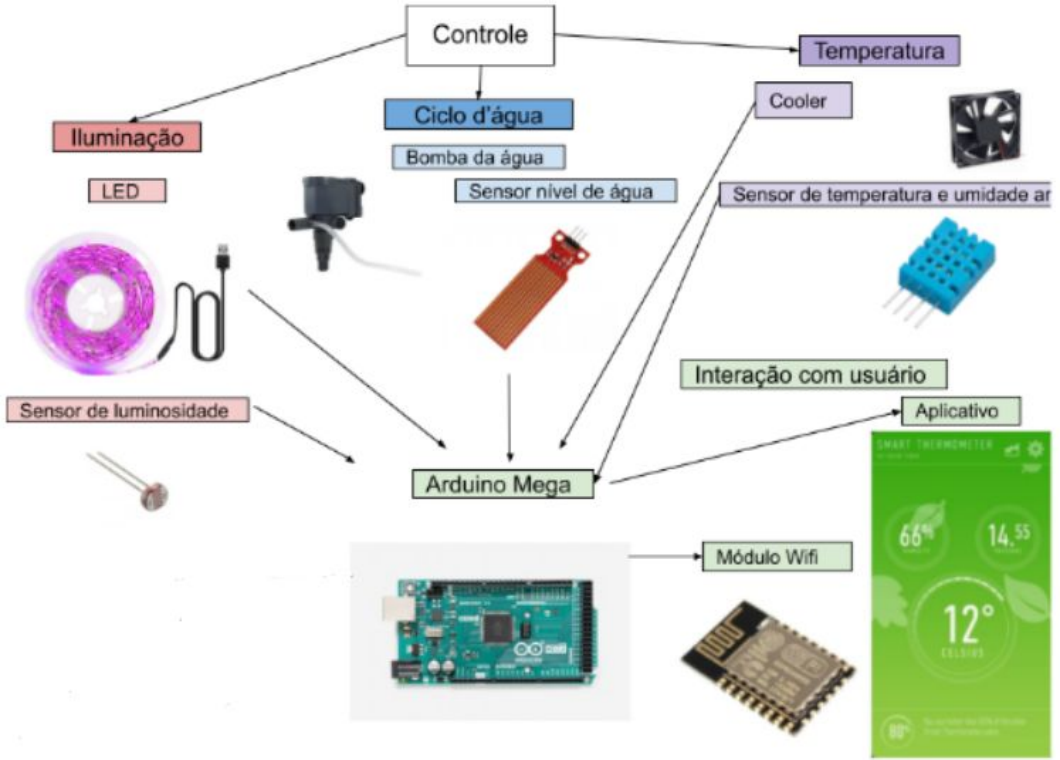
AeroFarms

INSTITUTO FEDERAL
Santa Catarina

CONCEIVE

- Sistemas a serem controlados:
  - Sistema de Iluminação
  - Sistema de Temperatura
  - Ciclo d'água
  - Conectividade e Comunicação

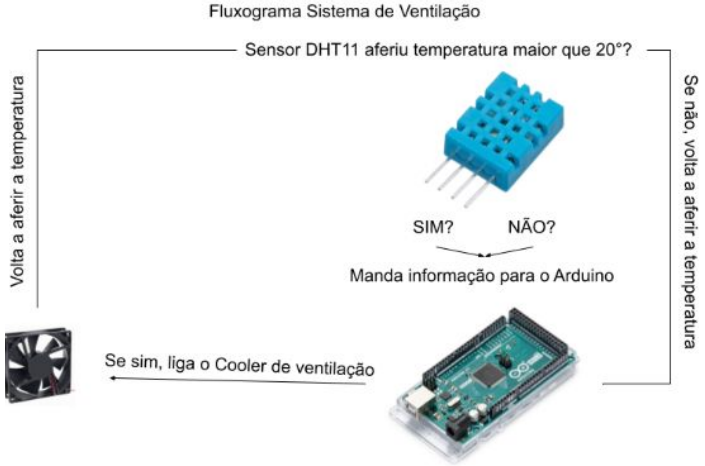




INSTITUTO FEDERAL
Santa Catarina

DESIGN

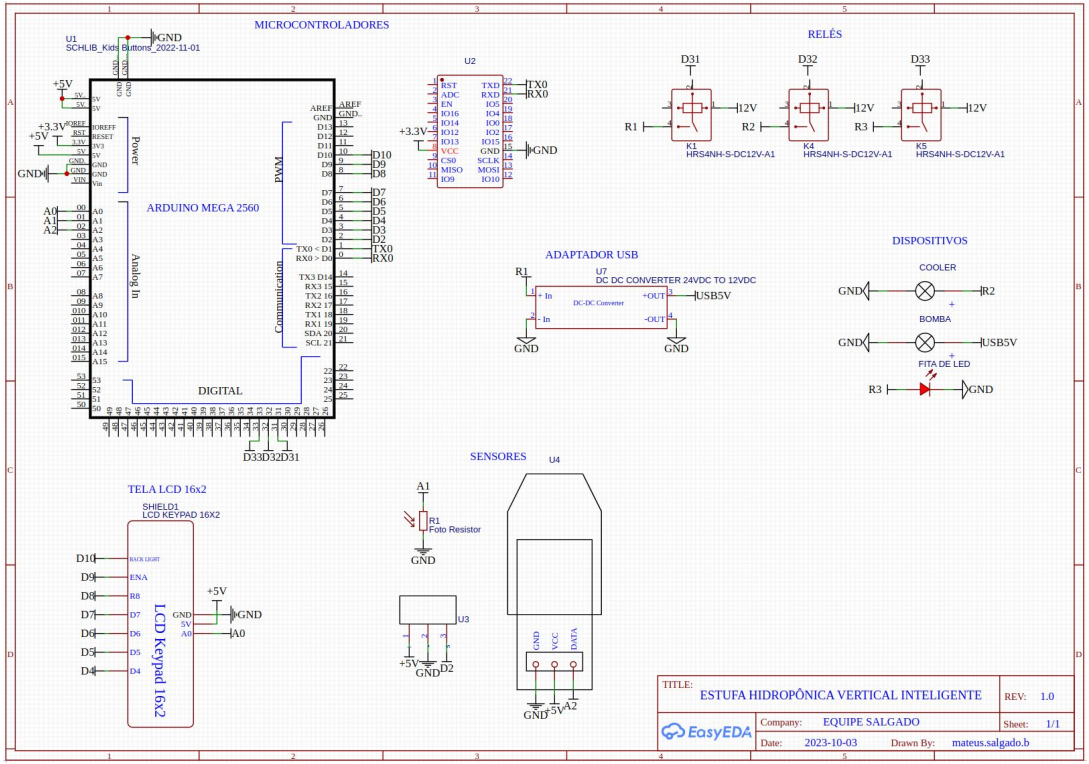
Diagrama de Funcionamento dos Sistemas

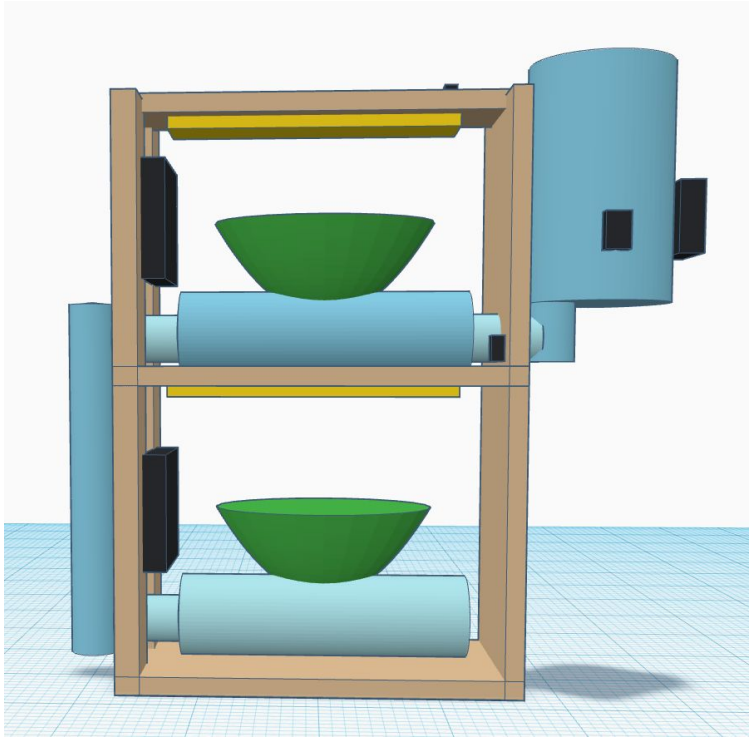


INSTITUTO FEDERAL
Santa Catarina

DESIGN

Plantas Elétricas e Mecânicas





INSTITUTO FEDERAL
Santa Catarina

DESIGN

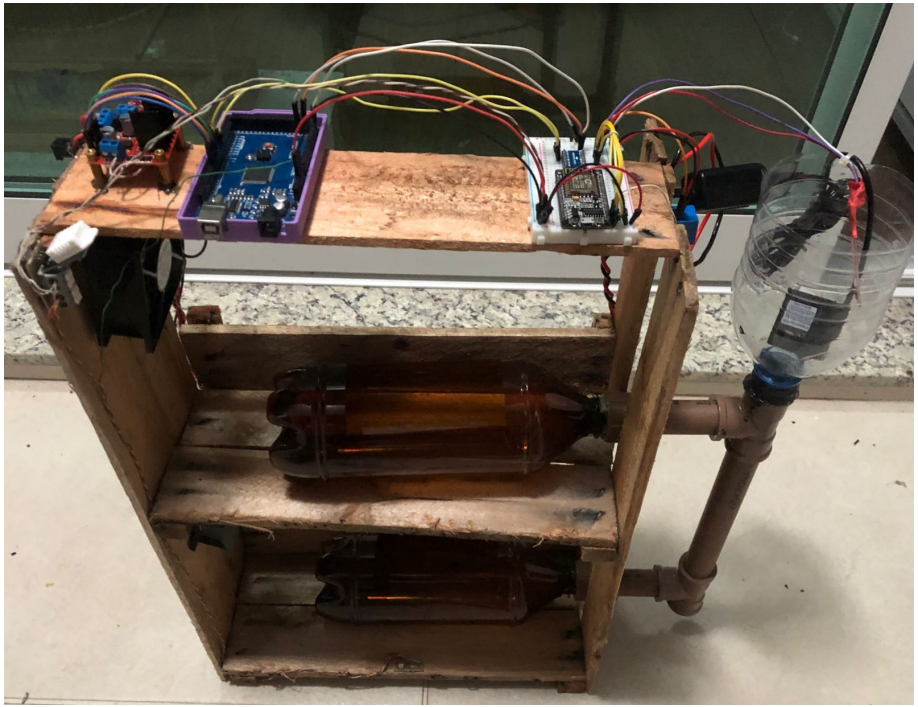
Aplicativo Blynk

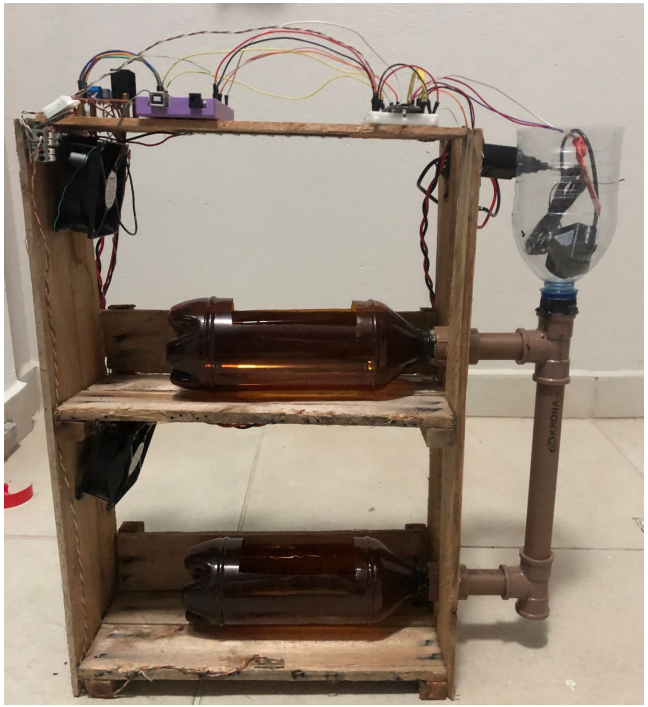
Id	Name	Alias	Color	Pin	Data Type	Units	Is Raw	Min	Max
1	pwmBOMBA	pwmBOMBA		V0	Integer		false	0	1
7	nivelValue	nivelValue		V3	Integer		false	0	1000
8	nivelTSH	nivelTSH		V4	Integer		false	0	1000
11	pwmLED	pwmLED		V8	Integer		false	0	100
6	luminosidadeValue	luminosidadeValue		V1	Integer		false	0	1000
2	luminosidadeTSH	luminosidadeTSH		V2	Integer		false	0	1000
3	temperaturaValue	temperaturaValue		V6	Integer	°C	false	0	50
9	temperaturaTSH	temperaturaTSH		V5	Integer		false	0	50
10	pwmCOOLER	pwmCOOLER		V7	Integer		false	0	255

INSTITUTO FEDERAL
Santa Catarina

IMPLEMENT

Parte Mecânica e Elétrica Instaladas





INSTITUTO FEDERAL
Santa Catarina

IMPLEMENT

Classe SensorController.h

estufa | Communication.cpp | Communication.h | DeviceController.cpp | DeviceController.h | SensorController.cpp | SensorController.h

```
#ifndef SensorController_h
#define SensorController_h

#include <DHT.h>

class SensorController {
private:
  const int LDRpin;
  const int T1592vcc;
  const int T1592pin;
  const int DHTpin;
  DHT dht;

public:
  SensorController(int ldrPin, int t1592vcc, int t1592pin, int dhtPin);
  struct SensorValues {
    int lighting;
    int level;
    int temperature;
    int humidity;
  };
  SensorValues readAllSensors(bool debug = false);

private:
  struct DHT22Value {
    int temperature;
    int humidity;
  };
  int readLDR(bool debug);
  int readT1592(bool debug);
  DHT22Value readDHT22(bool debug);
};

#endif
```

INSTITUTO FEDERAL
Santa Catarina

IMPLEMENT

Função readAllSensors()

estufa | Communication.cpp | Communication.h | DeviceController.cpp | DeviceController.h | SensorController.cpp | SensorController.h

```
#include "SensorController.h"

SensorController::SensorController(int ldrPin, int t1592vcc, int t1592pin, int dhtPin)
    : LDRpin(ldrPin), T1592vcc(t1592vcc), T1592pin(t1592pin), DHTpin(dhtPin), dht(dhtPin, DHT22) {
  pinMode(LDRpin, INPUT);
  pinMode(T1592vcc, OUTPUT);
  pinMode(T1592pin, INPUT);
  pinMode(DHTpin, INPUT);
  dht.begin();
}

SensorController::SensorValues SensorController::readAllSensors(bool debug) {
  SensorValues values;

  values.lighting = readLDR(debug);
  values.level = readT1592(debug);

  DHT22Value dhtData = readDHT22(debug);
  values.temperature = dhtData.temperature;
  values.humidity = dhtData.humidity;

  return values;
}
```

INSTITUTO FEDERAL
Santa Catarina

IMPLEMENT

Classe DeviceController.h

estufa | Communication.cpp | Communication.h | DeviceController.cpp | DeviceController.h | SensorController.cpp | SensorController.h

```
#ifndef DeviceController_h
#define DeviceController_h

#include <Arduino.h>

class DeviceController {
public:
  DeviceController(int coolerPin1, int coolerPin2, int coolerPWM, int ledPin1, int ledPin2, int ledPWM, int bombaPin);

  struct DeviceStatus {
    int led;
    int cooler;
    int bomba;
  };

  DeviceStatus statusDevice();
  void controlDevice(int device, int pwm);

private:
  int COOLERpin1, COOLERpin2, COOLERpwm, LEDpin1, LEDpin2, LEDpwm, BOMBApin;
};

#endif
```

INSTITUTO FEDERAL
Santa Catarina

IMPLEMENT

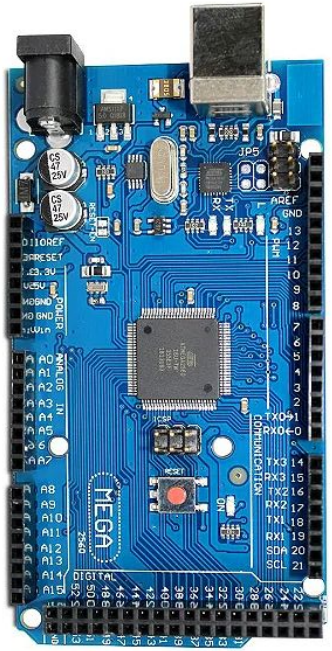
Função controlDevice()

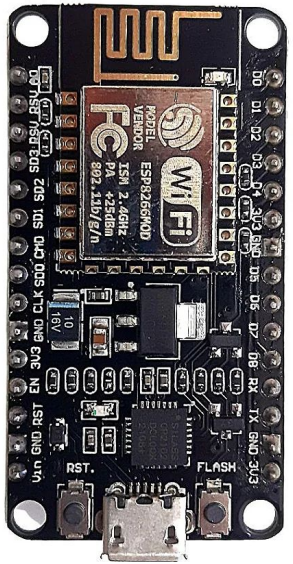
```
DeviceController::DeviceStatus DeviceController::statusDevice() {
  DeviceStatus retorno;

  retorno.cooler = analogRead(COOLERpwm);
  retorno.led = analogRead(LEDpwm);
  retorno.bomba = digitalRead(BOMBApin);

  return retorno;
}

void DeviceController::controlDevice(int device, int pwm) {
  switch (device) {
    // temperatura
    case 0:
      digitalWrite(COOLERpin1, HIGH);
      digitalWrite(COOLERpin2, LOW);
      analogWrite(COOLERpwm, pwm);
      break;

    // iluminacao
    case 1:
      digitalWrite(LEDpin1, HIGH);
      digitalWrite(LEDpin2, LOW);
      analogWrite(LEDpwm, pwm);
      break;

    // agua
    case 10:
      if (pwm > 0) {
        digitalWrite(BOMBApin, HIGH);
        }
      else{
        digitalWrite(BOMBApin, LOW);
        }
      break;

    default:
      break;
  }
}
```

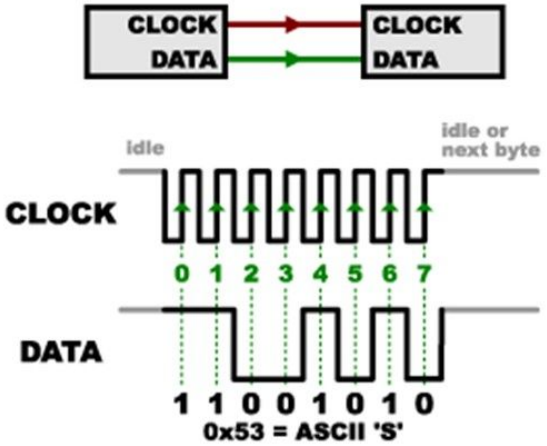
INSTITUTO FEDERAL
Santa Catarina

IMPLEMENT

Comunicação Serial







INSTITUTO FEDERAL
Santa Catarina

IMPLEMENT

PROTOCOLO DE COMUNICAÇÃO

s:00:000:0000:000:e

A interpretação deste pacote é a seguinte:

- start: indica o início do pacote.
- device: representa o dispositivo (cooler, LED, bomba).
- pwm: refere-se à largura de pulso modificada, variando de 0 a 255.
- sensor: indica o valor do sensor, variando conforme a aplicação.
- threshold: representa o valor de threshold, variando conforme a aplicação.
- e: indica o final do pacote.

INSTITUTO FEDERAL
Santa Catarina

IMPLEMENT

Classe Communication.h

estufa | Communication.cpp | Communication.h | DeviceController.cpp | DeviceController.h | SensorController.cpp | SensorController.h

```
#ifndef Communication_h
#define Communication_h

#include <Arduino.h>

class Communication {
public:
  struct Protocol {
    int device;
    int pwm;
    int sensor;
    int tsh;
  };

  Communication();

  void writeSerial(const Protocol& message);
  Protocol readSerial(bool debug);
  void printSerial(const char message[18]);
};

#endif
```

INSTITUTO FEDERAL
Santa Catarina

IMPLEMENT

Função writeSerial()

estufa | Communication.cpp | Communication.h | DeviceController.cpp | DeviceController.h | SensorController.cpp | SensorController.h

```
#include "Communication.h"

Communication::Communication() {}

void Communication::writeSerial(const Protocol& message) {
  char device[3];
  char pwmvalue[4];
  char sensorvalue[5];
  char tshvalue[4];

  sprintf(device, "%02d", message.device);
  sprintf(pwmvalue, "%03d", message.pwm);
  sprintf(sensorvalue, "%04d", message.sensor);
  sprintf(tshvalue, "%03d", message.tsh);

  Serial.write('s');
  Serial.write(':');
  Serial.write(device[0]);
  Serial.write(device[1]);
  Serial.write(':');
  Serial.write(pwmvalue[0]);
  Serial.write(pwmvalue[1]);
  Serial.write(pwmvalue[2]);
  Serial.write(':');
  Serial.write(sensorvalue[0]);
  Serial.write(sensorvalue[1]);
  Serial.write(sensorvalue[2]);
  Serial.write(sensorvalue[3]);
  Serial.write(':');
  Serial.write(tshvalue[0]);
  Serial.write(tshvalue[1]);
  Serial.write(tshvalue[2]);
  Serial.write(':');
  Serial.write('e');
}
```

INSTITUTO FEDERAL
Santa Catarina

IMPLEMENT

estufa | Communication.cpp | Communication.h | DeviceController.cpp | DeviceController.h | SensorController.cpp | SensorController.h

```
Communication communication;

struct Sistema {
  int pwm;
  int sensor;
  int tsh;
};

struct Estufa {
  Sistema luminosidade;
  Sistema temperatura;
  Sistema nivel;
};

void loop(){
  estufa = recieveProtocol(estufa, 1);

  static unsigned long lastTime1 = 0;
  static unsigned long lastTime2 = 0;
  unsigned long currentTime = millis();
  if (currentTime - lastTime2 >= 10000) {
    sendStatus(estufa, 1);
    lastTime2 = currentTime;
  }
}
```

INSTITUTO FEDERAL
Santa Catarina

IMPLEMENT

```
Estufa recieveProtocol(Estufa estufa, bool debug) {
  Communication::Protocol message = communication.readSerial(debug);
  delay(8000);
  deviceController.controlDevice(message.device, message.pwm);

  switch (message.device) {
    // temperatura
    case 0:
      estufa.temperatura.tsh = message.tsh;
      estufa.temperatura.pwm = message.pwm;

      if(estufa.temperatura.tsh < estufa.temperatura.sensor){
        estufa.temperatura.pwm = 130;
        deviceController.controlDevice(message.device, estufa.temperatura.pwm);
      }

      break;

    // iluminacao
    case 1:
      estufa.luminosidade.tsh = message.tsh;
      estufa.luminosidade.pwm = message.pwm;

      if(estufa.luminosidade.tsh > estufa.luminosidade.sensor){
        estufa.luminosidade.pwm = 100;
        deviceController.controlDevice(message.device, estufa.luminosidade.pwm);
      }

      break;

    // agua
    case 10:
      estufa.nivel.tsh = message.tsh;
      estufa.nivel.pwm = message.pwm;

      if(estufa.nivel.tsh < estufa.nivel.sensor){
        estufa.nivel.pwm = 100;
        deviceController.controlDevice(message.device, estufa.nivel.pwm);
      }

      break;

    default:
      break;
  }

  return estufa;
}
```

```
void sendStatus(Estufa estufa, bool debug) {
  SensorController::SensorValues sensor = sensorController.readAllSensors(debug);

  message.device = 0;
  message.pwm = estufa.temperatura.pwm;
  message.sensor = sensor.temperature;
  message.tsh = estufa.temperatura.tsh;

  communication.writeSerial(message);

  message.device = 1;
  message.pwm = estufa.luminosidade.pwm;
  message.sensor = sensor.lighting;
  message.tsh = estufa.luminosidade.tsh;

  communication.writeSerial(message);

  message.device = 10;
  message.pwm = estufa.nivel.pwm;
  message.sensor = sensor.level;
  message.tsh = estufa.nivel.tsh;

  communication.writeSerial(message);
}
```

INSTITUTO FEDERAL
Santa Catarina

IMPLEMENT

blynk | Communication.cpp | Communication.h

```
/* Fill-in information from Blynk Device Info here */
#define BLYNK_TEMPLATE_ID "TMPL2hDK5YqQo"
#define BLYNK_TEMPLATE_NAME "Teste"
#define BLYNK_AUTH_TOKEN "L-RXKJ_z_vb6QLuERlDHT_Zff05wGCAT"
#define BLYNK_PRINT Serial


#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

#include "Communication.h"
Communication communication;

// WiFi credentials.
char ssid[] = "arduino";
char pass[] = "esp-8266";

BlynkTimer timer;

struct Sistema {
  int pwm;
  int sensor;
  int tsh;
};

struct Estufa {
  Sistema luminosidade;
  Sistema temperatura;
  Sistema nivel;
};

Estufa estufa;
Communication::Protocol message;
```

```
void recieveProtocol() {
  Communication::Protocol message = communication.readSerial(0);

  switch (message.device) {
    // temperatura
    case 0:
      Blynk.virtualWrite(V7, message.pwm);
      Blynk.virtualWrite(V6, message.sensor);
      Blynk.virtualWrite(V5, message.tsh);

      estufa.temperatura.sensor = message.sensor;
      estufa.temperatura.pwm = message.pwm;

      break;

    // iluminacao
    case 1:
      Blynk.virtualWrite(V8, message.pwm);
      Blynk.virtualWrite(V1, message.sensor);
      Blynk.virtualWrite(V2, message.tsh);

      estufa.luminosidade.sensor = message.sensor;
      estufa.luminosidade.pwm = message.pwm;

      break;

    // agua
    case 10:
      Blynk.virtualWrite(V0, message.pwm);
      Blynk.virtualWrite(V3, message.sensor);
      Blynk.virtualWrite(V4, message.tsh);

      estufa.nivel.sensor = message.sensor;
      estufa.luminosidade.pwm = message.pwm;

      break;

    default:
      break;
  }
}
```

```
BLYNK_WRITE(V7)
{
  estufa.temperatura.pwm = param.asInt();

  message.device = 0;
  message.pwm = estufa.temperatura.pwm;
  message.sensor = estufa.temperatura.sensor;
  message.tsh = estufa.temperatura.tsh;

  communication.writeSerial(message);
}
```

INSTITUTO FEDERAL
Santa Catarina

OPERATE

```
Luminosidade LDR: 361
Nível de água: 17
Umidade: 90 %
Temperatura: 23 °C

s:00:000:0023:000:es:01:000:0361:000:es:10:000:0017:000:e
```

```
Device Value: 0
PWM Value: 180
Sensor Value: 23
TSH Value: 25
```

```
ZE§§X§l§Y0§§N§hM§/H§hG:§§[65] Connecting to @fbarbozac
[6389] Connected to WiFi
[6390] IP: 192.168.1.151
[6390]
    ___  __          __
   / _ )/ /_ _____  / /__
  / _  / / // / _ \/  '_/
 /____/_/\_, /_//_/_/\_\
        /___/ v1.3.2 on ESP8266

 #StandWithUkraine    https://bit.ly/swua


[6516] Connecting to blynk.cloud:80
[12554] Connecting to blynk.cloud:8080
[18554] Connecting to blynk.cloud:80
[19073] Ready (ping: 195ms).
[19140] Connecting to @fbarbozac
[19140] Connected to WiFi
[19140] IP: 192.168.1.151
[19140]
    ___  __          __
   / _ )/ /_ _____  / /__
  / _  / / // / _ \/  '_/
 /____/_/\_, /_//_/_/\_\
        /___/ v1.3.2 on ESP8266

 #StandWithUkraine    https://bit.ly/swua


[19305] Connecting to blynk.cloud:80
[19641] Ready (ping: 149ms).
s:00:104:0000:000:es:01:000:0000:535:es:10:000:0000:539:e
```