UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Edital n°05/2023
Programa de Extensão Tecnológica

Curso de Capacitação:
Internet das Coisas Aplicada a Agropecuária de Precisão

Código do processo: ARC-0423-5.03/23

Victor Medeiros
victor.wanderley@ufrpe.br

FACEPE Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

SEMINE AGRICULTURA IRRIGADA INTELIGENTE

SMART RURAL

Juá Labs

Quem sou eu?

Victor Medeiros

Arquitetura de Computadores
Sistemas Embarcados
Internet das Coisas
Computação em Nuvem
Computação de Alto Desempenho

Juá Labs

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

O curso

Internet das Coisas Aplicada a Agropecuária de Precisão

- Híbrido
- 50 vagas
- 5 semanas de duração
- 30 horas de carga horária
- Atividades práticas em todas as semanas
- Utilizaremos o Google Classroom como plataforma para acompanhamento do curso e comunicação

https://classroom.google.com/c/NjI4OTM1ODUyMjly?cjc=no3nql6

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

O curso

- **Cronograma**
  - **Semana 1 (6h):** Introdução à IoT, plataformas abertas e linguagem C no desenvolvimento IoT
  - **Semana 2 (6h):** Conceitos básicos de eletrônica aplicados ao desenvolvimento de sistemas IoT para agropecuária de precisão
  - **Semana 3 (6h):** Sensores e atuadores comumente utilizados na agropecuária de precisão
  - **Semana 4 (6h):** Redes de comunicação e infraestrutura de nuvem para sistemas IoT aplicados à agropecuária de precisão
  - **Semana 5 (6h):** Prototipação de um projeto telemétrico para a agropecuária de precisão.

O curso

- **Ementa:**
  - Introdução à Internet das Coisas (IoT)
  - Sensores e Atuadores para a IoT
  - Plataformas abertas para IoT
  - Infraestrutura de Comunicação para a IoT
  - Infraestrutura de Nuvem para Armazenamento, Processamento e Visualização de dados
  - Estudos de Caso e Aplicações Práticas de IoT
  - Projeto Prático.

Atividades de Extensão

- **Durante 3 meses, 10 extensionistas bolsistas atuarão no desenvolvimento e aperfeiçoamento de soluções tecnológicas para as startups envolvidas no projeto.**

SEMINE
AGRICULTURA IRRIGADA INTELIGENTE

SMART RURAL


Vamos lá?
victor.wanderley@ufrpe.br
FOTO: MARCOS SANTOS / USP IMAGENS

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Internet das Coisas Aplicada a Agropecuária de Precisão

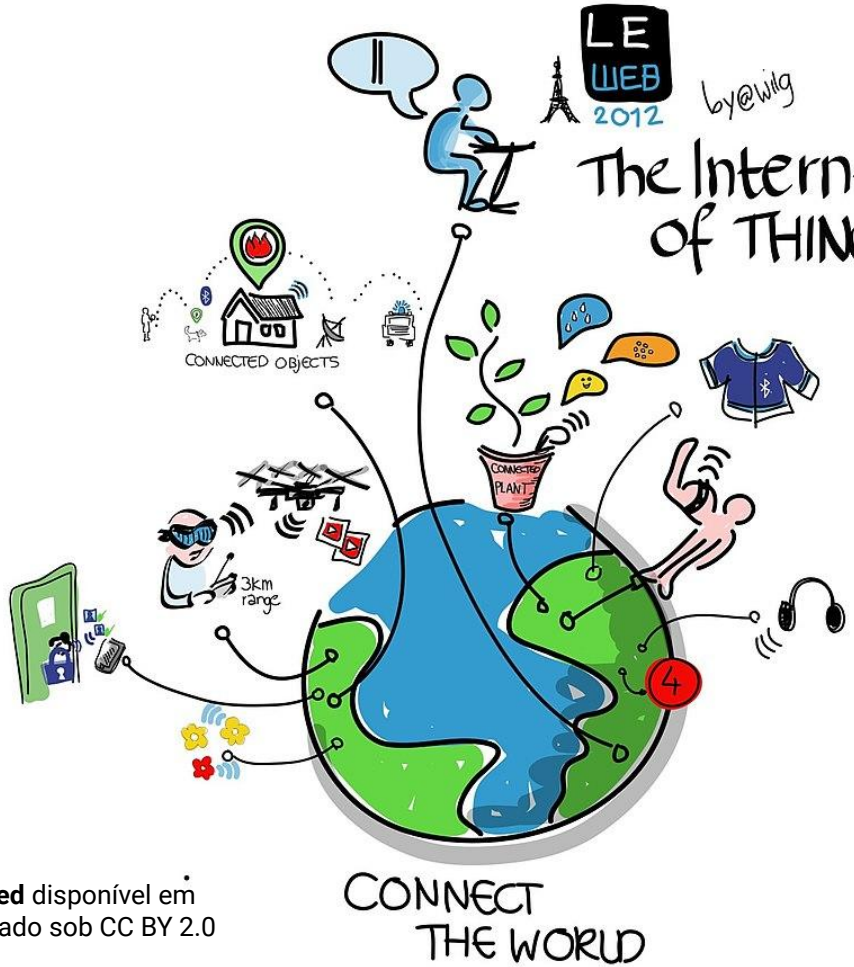
Alguns conceitos importantes

FACEPE Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

SEMINE AGRICULTURA IRRIGADA INTELIGENTE

SMART RURAL

Juá Labs



Embedded System? IoT? CPS?

Embarcado, IoT ou CPS?

AS REVOLUÇÕES INDUSTRIAIS

1ª Mecanização

Teve início no fim do século XVIII e consistiu na mecanização da produção usando água e energia a vapor.

2ª Eletricidade

Teve início um século mais tarde e consistiu no aumento da produção através das linhas de montagem e do uso de energia elétrica.

3ª Automação

Teve início em meados do século XX e consistiu na introdução da automação e robótica aproveitando-se dos avanços da eletrônica.

4ª Grande integração de Tecnologias da Informação

Fontes:
https://www.ibm.com/topics/industry-4-0

Li, K., Zhou, T. & Liu, Bh. **Internet-based intelligent and sustainable manufacturing: developments and challenges**. Int J Adv Manuf Technol 108, 1767–1791 (2020). https://doi.org/10.1007/s00170-020-05445-0

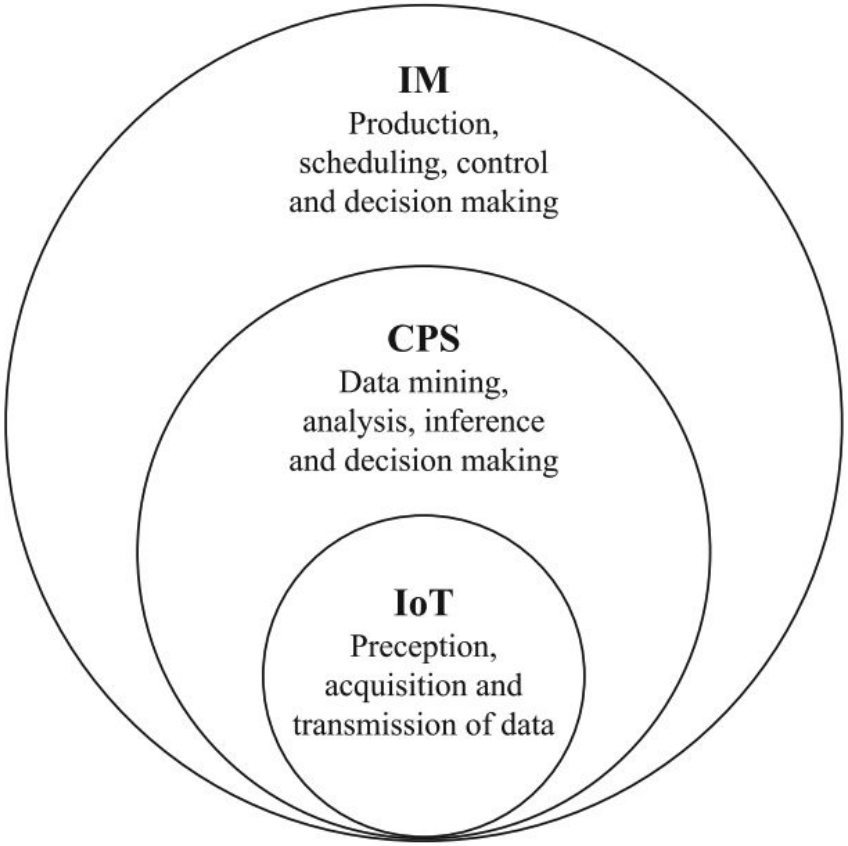
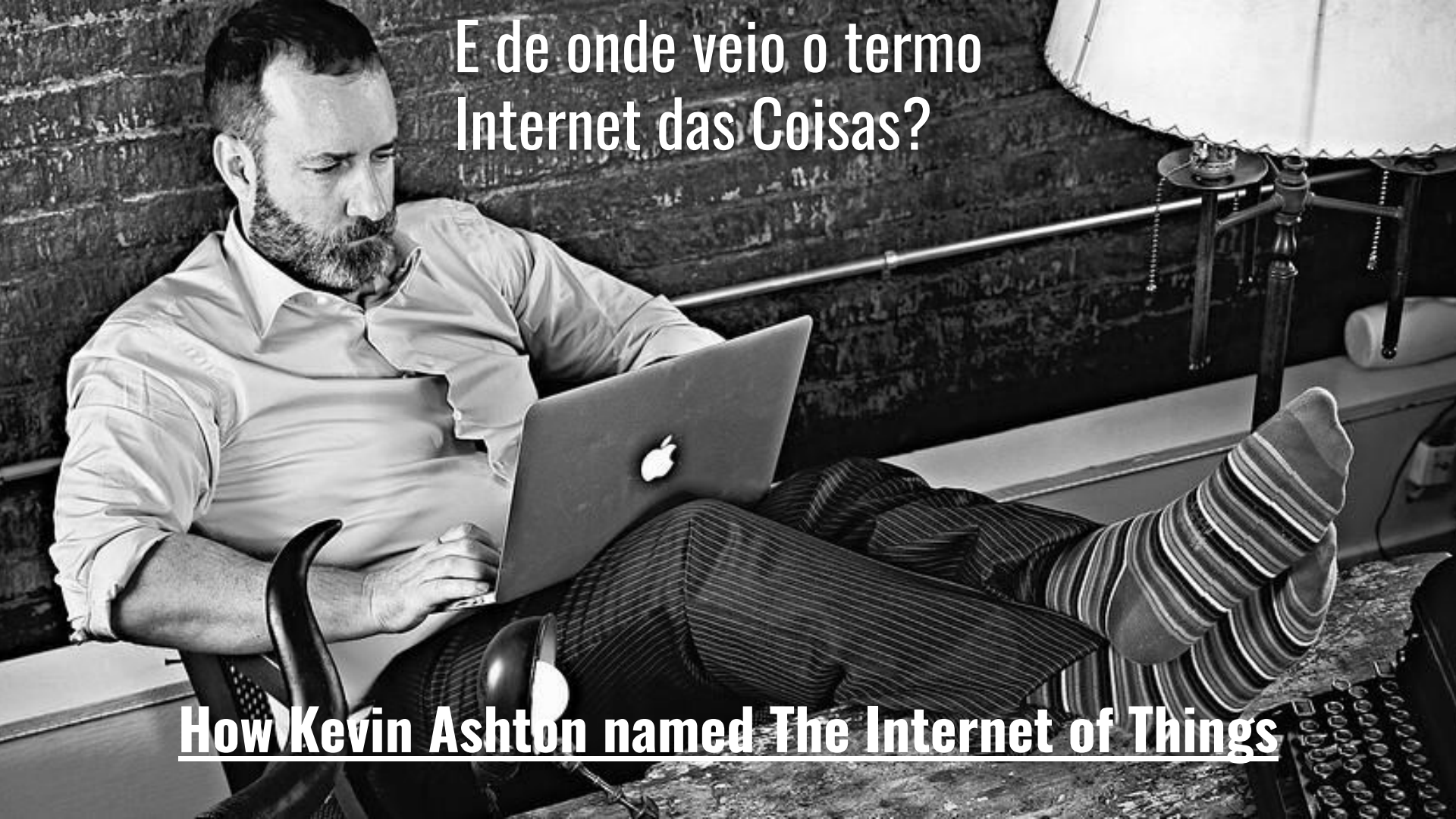


Fig. 4 The proposed relationships between IM, CPS, and IoT

E de onde veio o termo Internet das Coisas?



How Kevin Ashton named The Internet of Things

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Internet das Coisas Aplicada a Agropecuária de Precisão

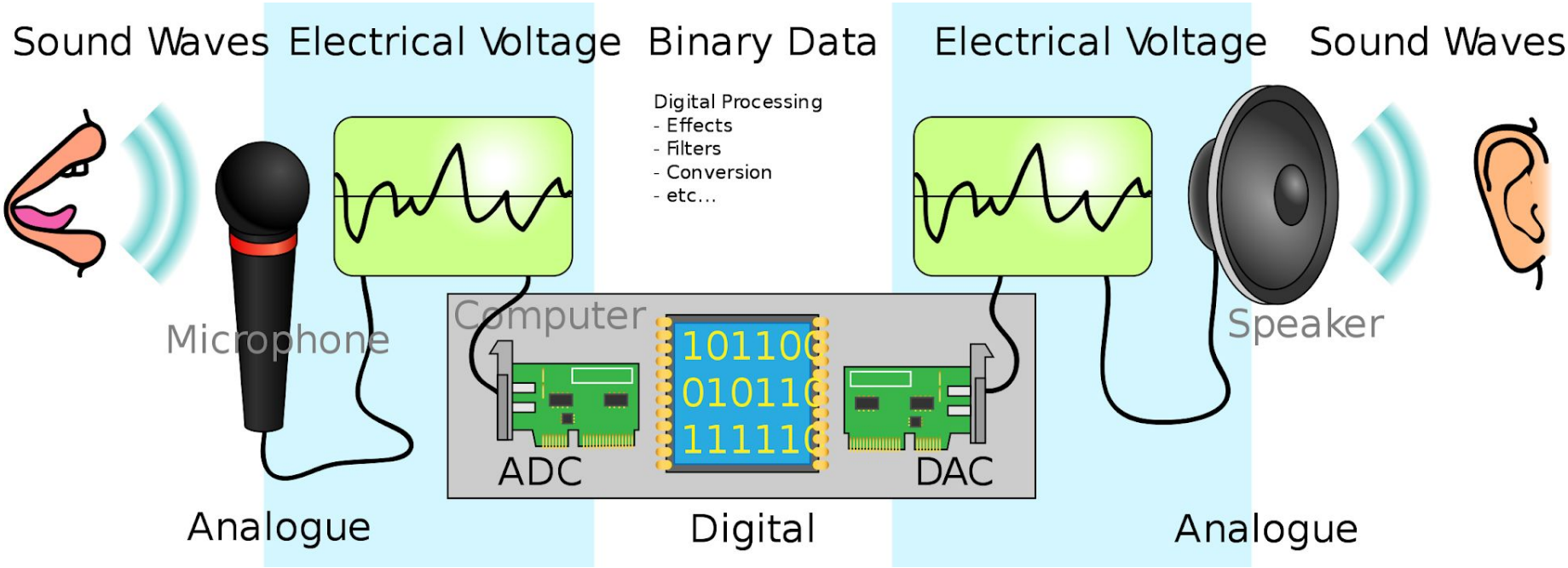
Analógico vs. Digital

FACEPE Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

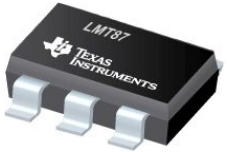
SEMINE AGRICULTURA IRRIGADA INTELIGENTE

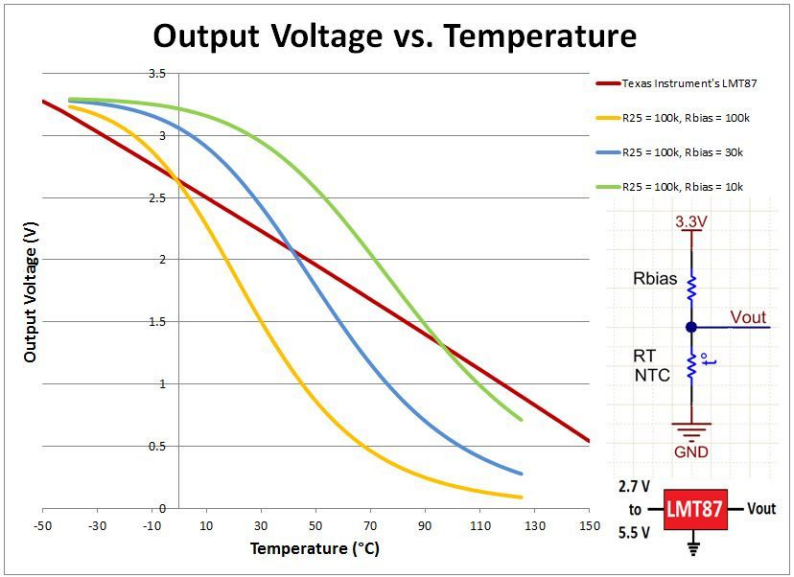
SMART RURAL

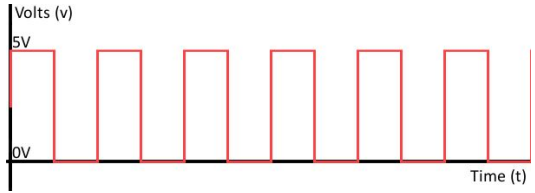
Juá Labs



analógico vs. digital

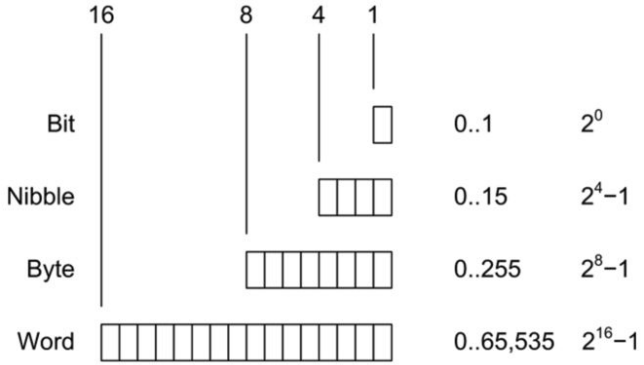






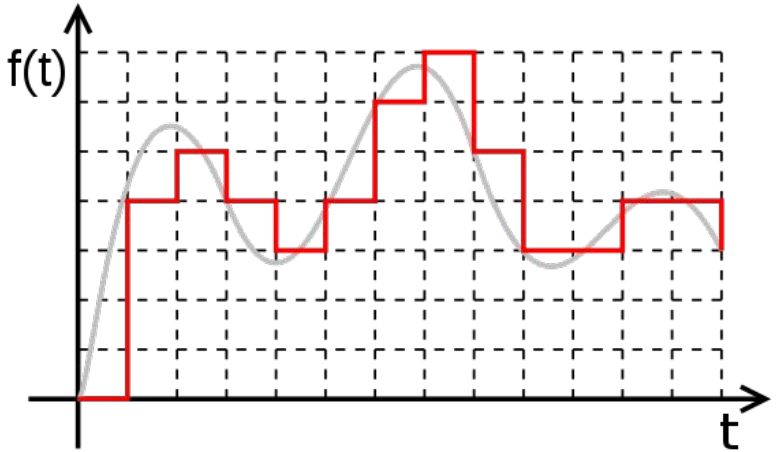
dados digitais

- dados digitais são valores binários codificados na forma de *bits* os quais podem ser manipulados diretamente pelo computador.
- dados digitais são discretos e um único *bit* pode assumir apenas dois valores: 1 ou 0, ligado ou desligado, verdadeiro ou falso.
- o número de *bits* utilizados por um dado digital determina o número de valores que podem ser representados.

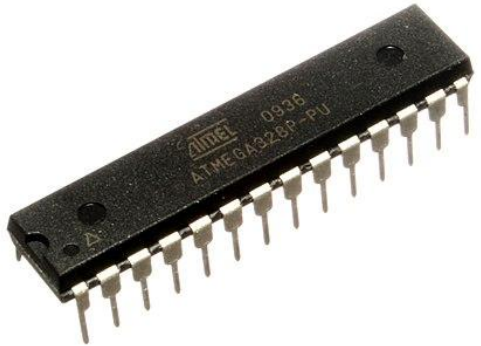


dados analógicos

- dados analógicos precisam ser convertidos em dados digitais para serem manipulados pelo computador.
- **Conversores A/D** são utilizados para isso.
- o número de *bits* do conversor A/D determina a resolução do conversor.



O módulo ESP32 WROOM dispõe de um conversor A/D de 16 canais com 12-bits de resolução.



O ATMEGA328P dispõe de um conversor A/D de 6 canais com 10-bits de resolução

O que isso significa?

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Internet das Coisas Aplicada a Agropecuária de Precisão

Sistemas de Controle

FACEPE Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

SEMINE AGRICULTURA IRRIGADA INTELIGENTE

SMART RURAL

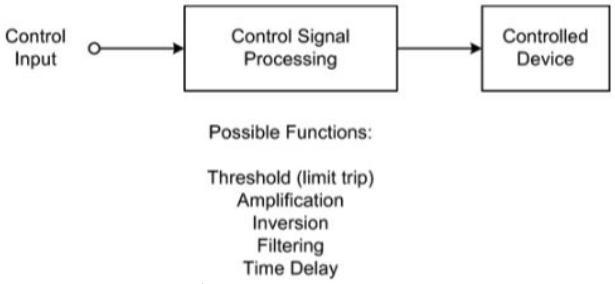
Juá Labs

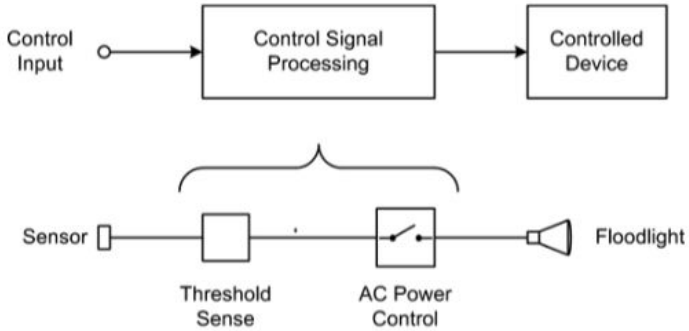
controle

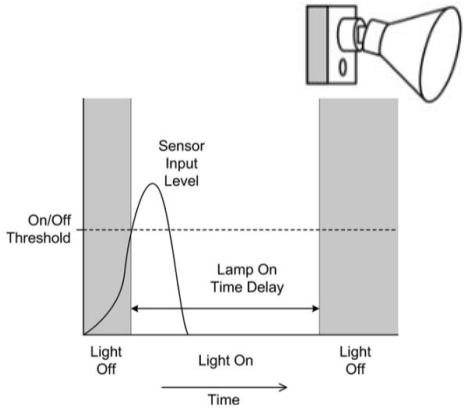
- sistemas IoT envolvem, além da comunicação, aquisição de dados e controle
- a aquisição realiza o sensoriamento do ambiente e gera dados de entrada
- o controle utiliza estes dados para realizar ações no mundo físico
- os sistemas de controle podem ser:
  - controle de malha aberta (open-loop control)
  - controle de malha fechada (closed-loop control)
  - controle sequencial

controle de malha aberta

- o controlador em um sistema de malha aberta é "cego"
- o dispositivo controlado poder ser: um motor elétrico; uma lâmpada; um ventilador; ou uma válvula.



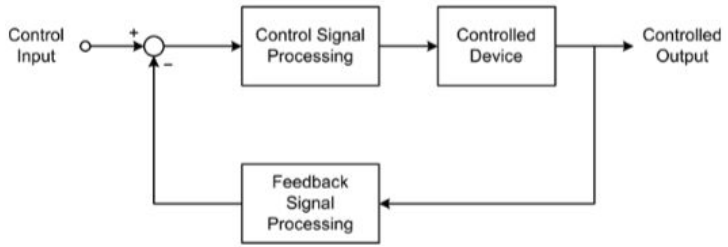


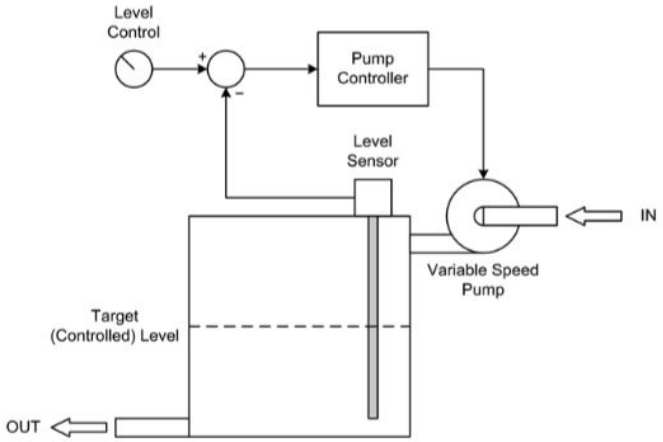


exemplo de um refletor de segurança

controle de malha fechada

- o controlador recebe *feedback* de suas ações
- o o sinal de *feedback* é adicionado ao sinal de entrada no nó de soma.
- o resultado recebe o nome de erro de controle

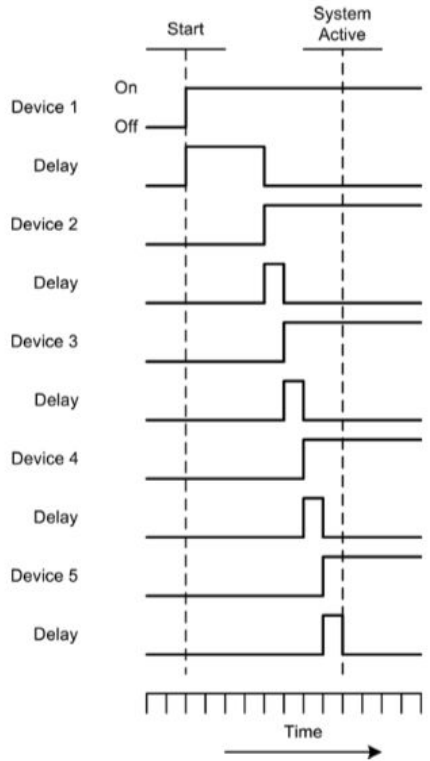




exemplo de um controlador de nível de fluído

controle sequencial

- em um sistema de controle sequencial uma série de diferentes eventos acontecem um após o outro
- o término de um evento na sequência gera o sinal para o próximo evento ter início
- exemplos de sistemas sequenciais:
  - máquinas de lavar
  - sinais de trânsito
  - elevadores



UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Internet das Coisas Aplicada a Agropecuária de Precisão

Plataformas de Software e Hardware Abertas na IoT

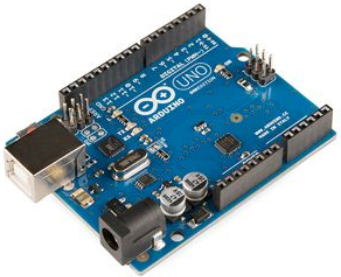
FACEPE
Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

SEMINE
AGRICULTURA IRRIGADA INTELIGENTE

SMART RURAL

Juá Labs

plataformas de prototipação em Internet das Coisas



histórico do Arduino

- Da esquerda pra direita:
  - David Cuartielles
  - Gianluca Martino
  - Tom Igoe
  - David Mellis
  - Massimo Banzi

histórico do Arduino

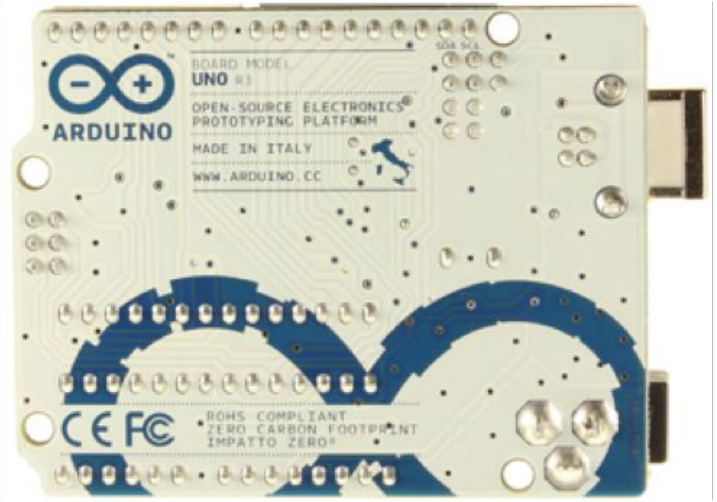
- Banzi, professor na **Interaction Design Institute Ivrea**, precisava de um hardware fácil de usar;
- Em 2005, foi criado o primeiro protótipo do que viria a se tornar o Arduino;
- Banzi e sua equipe resolveram adotar a filosofia **open-source**, pois acreditavam que isso ajudaria a atingir um maior número de pessoas, e também para tornar o projeto "auto-sustentável";
- Queriam que a placa fosse barata, porém estilosa!
- Mas e o nome?!?!

histórico do Arduino

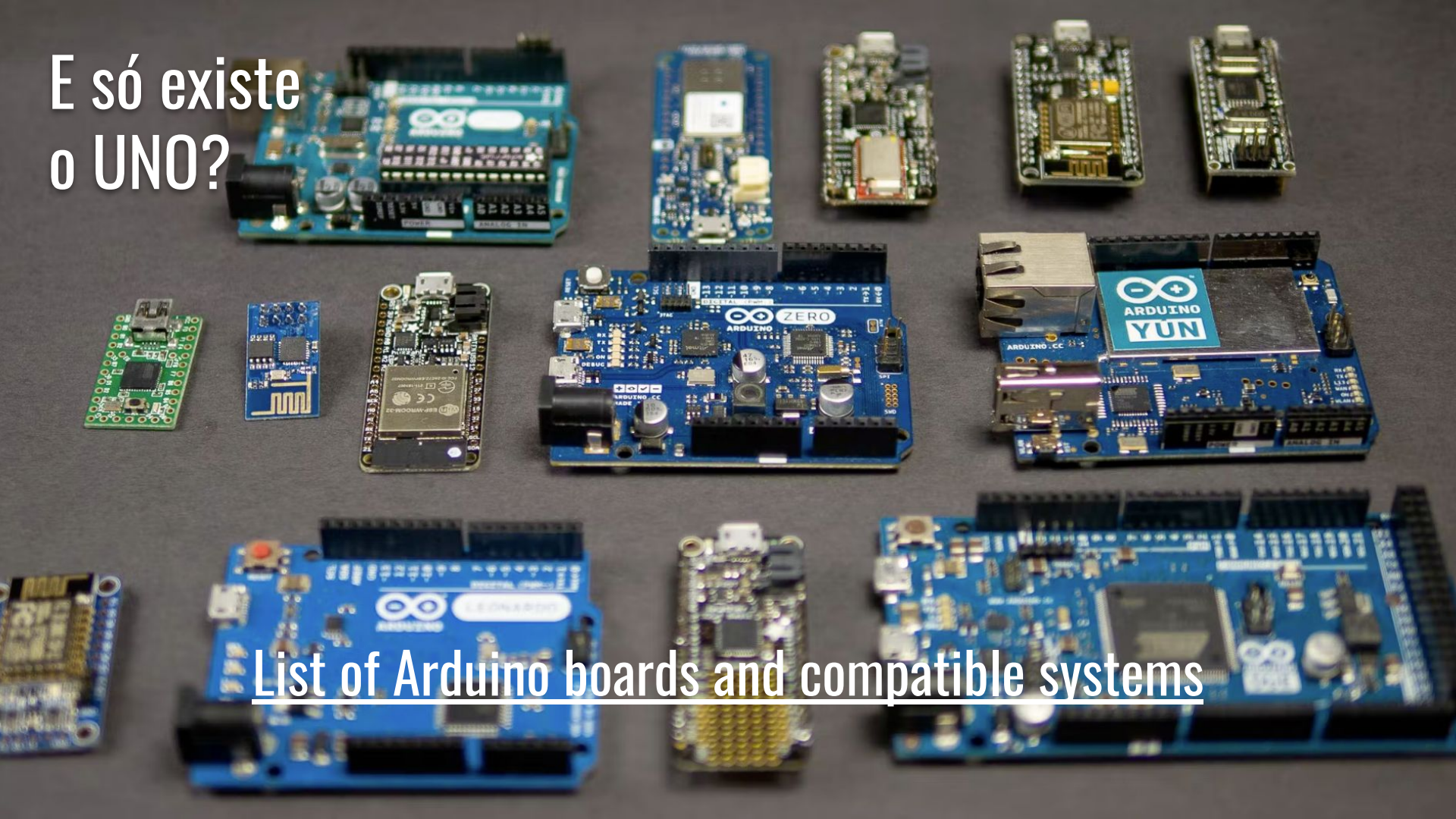
Arduino d'Ivrea
(Civica raccolta stampe - Milano)

Arduino UNO


DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN

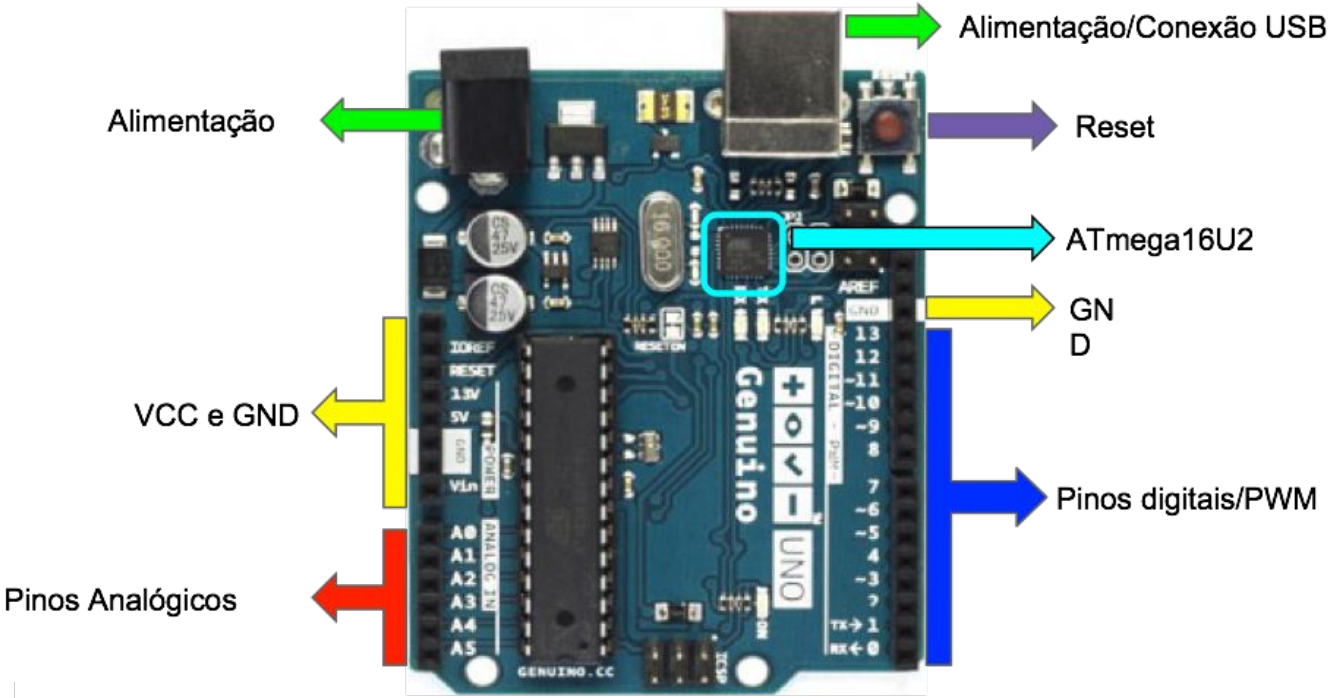

ARDUINO
BOARD MODEL
UNO R3
OPEN-SOURCE ELECTRONICS PROTOTYPING PLATFORM
MADE IN ITALY
WWW.ARDUINO.CC
ROHS COMPLIANT
ZERO CARBON FOOTPRINT
IMPATTO ZERO

E só existe o UNO?



List of Arduino boards and compatible systems

voltando ao UNO...



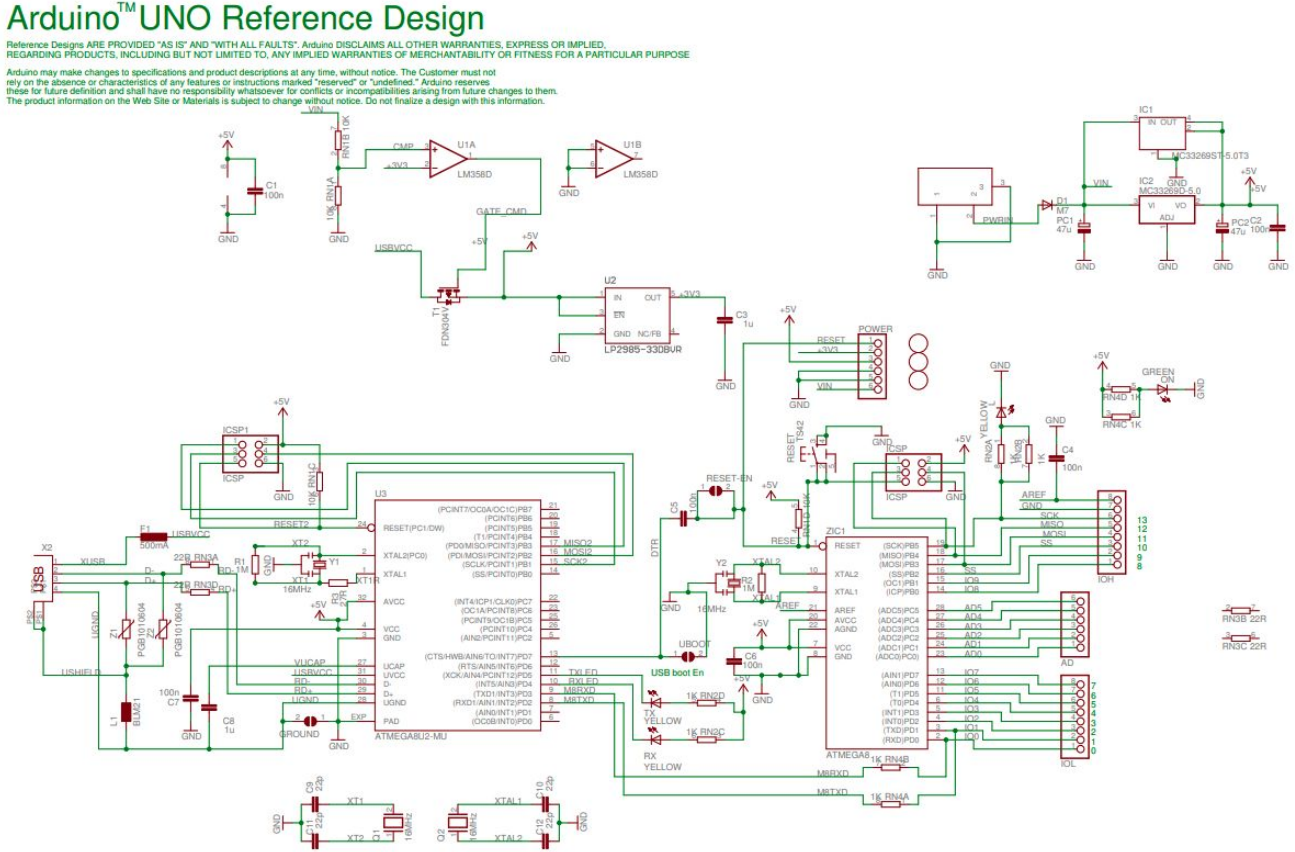
especificações técnicas...

• Microcontroller	=	ATmega328P
• Operating Voltage	=	5V
• Input Voltage (recommended)	=	7-12V
• Input Voltage (limit)	=	6-20V
• Digital I/O Pins	=	14 (of which 6 provide PWM output)
• PWM Digital I/O Pins	=	6
• Analog Input Pins	=	6
• DC Current per I/O Pin	=	20 mA
• DC Current for 3.3V Pin	=	50 mA
• Flash Memory	=	32 KB (ATmega328P) of which 0.5 KB used by bootloader
• SRAM	=	2 KB (ATmega328P)
• EEPROM	=	1 KB (ATmega328P)
• Clock Speed	=	16 MHz
• Length	=	68.6 mm
• Width	=	53.4 mm
• Weight	=	25 g

Bootloader, Firmware e *Application code*

- Bootloader:
  - Termo genérico para um software que permite a programação de uma memória não volátil (*flash* ou EPPROM). No caso do Arduino, é o firmware gravado no microcontrolador, e é executado quando o usuário clica em 'upload' na IDE oficial;
- Firmware:
  - Código baixo-nível (*low-level*). Dá suporte ao funcionamento do Application code. Ele lida com os componentes de hardware a fim de executar as tarefas definidas pelo usuário;
- Application code:
  - O código que o usuário vai desenvolver, para seu respectivo projeto, e gravar no microcontrolador. A linguagem "Arduino";

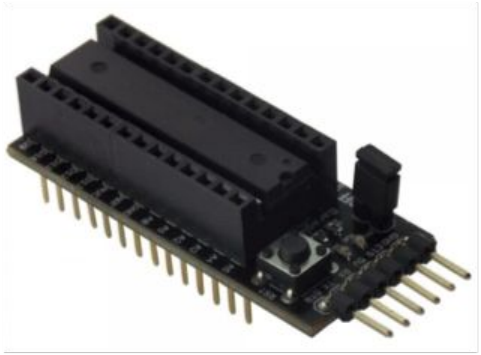
open hardware
não é da boca
pra fora!!!



Arduino UNO reference design

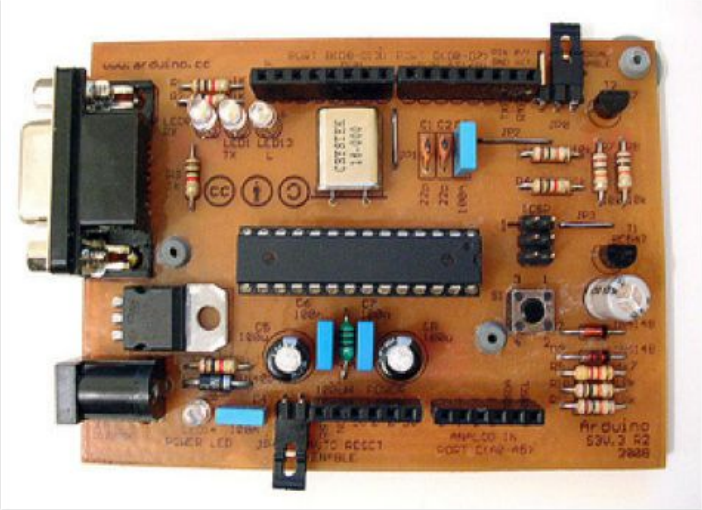
versões brasileiras

BlackBoard - Robocore

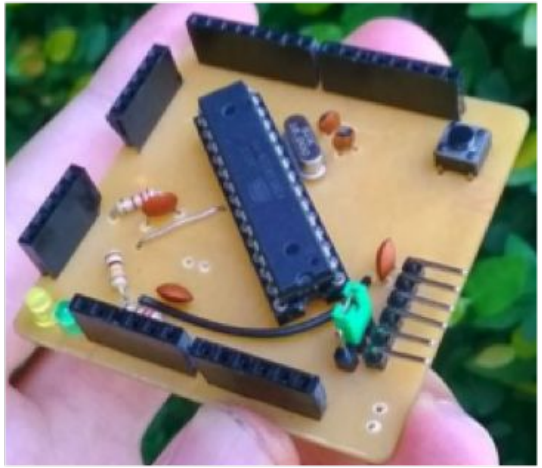


Garagino - Laboratório de Garagem

versões brasileiras



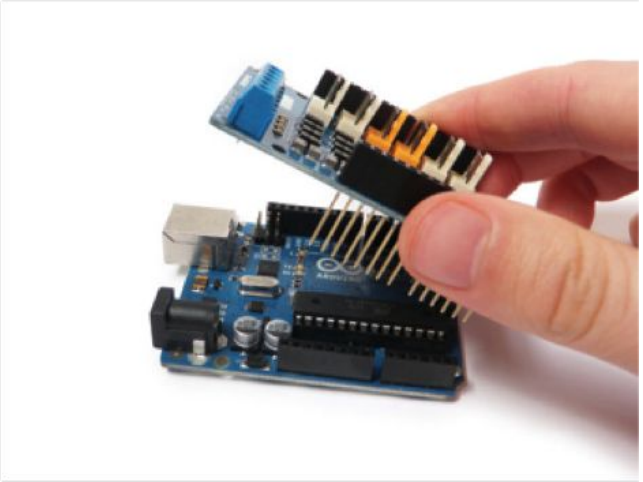
Severino



Marminino

shields

- Shields são placas empilháveis sobre o Arduino que trazem novas funcionalidades, como conexão com a internet;



shields

- O uso dos shields é feito por meio de bibliotecas

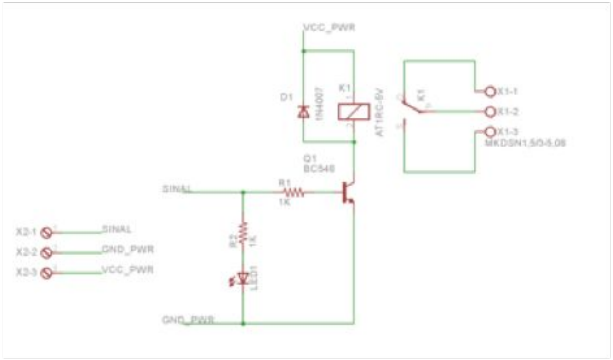
Ethernet

WiFi

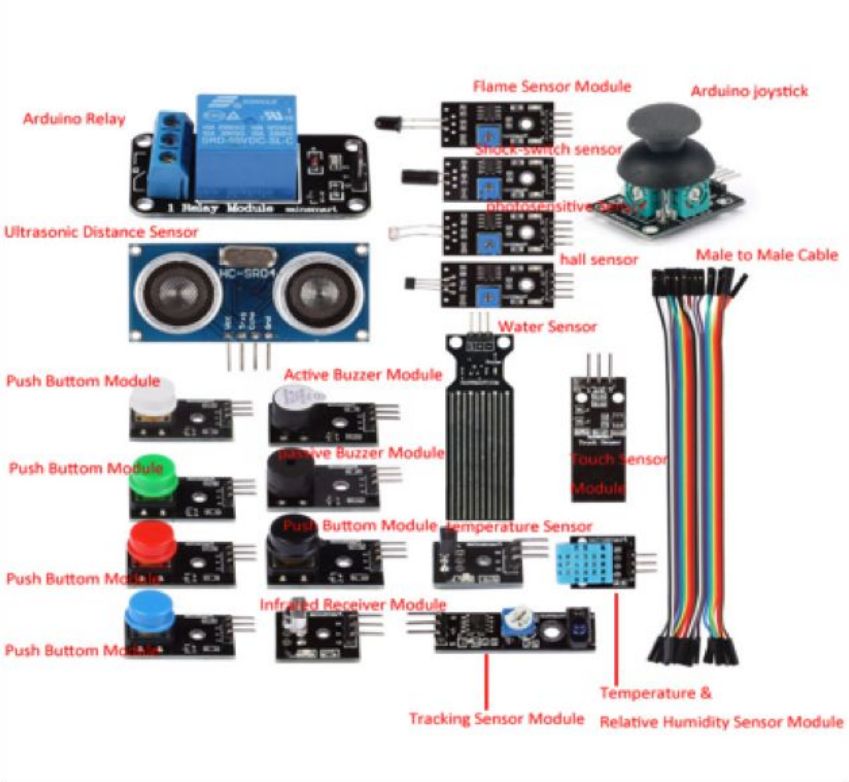
Motor

módulos

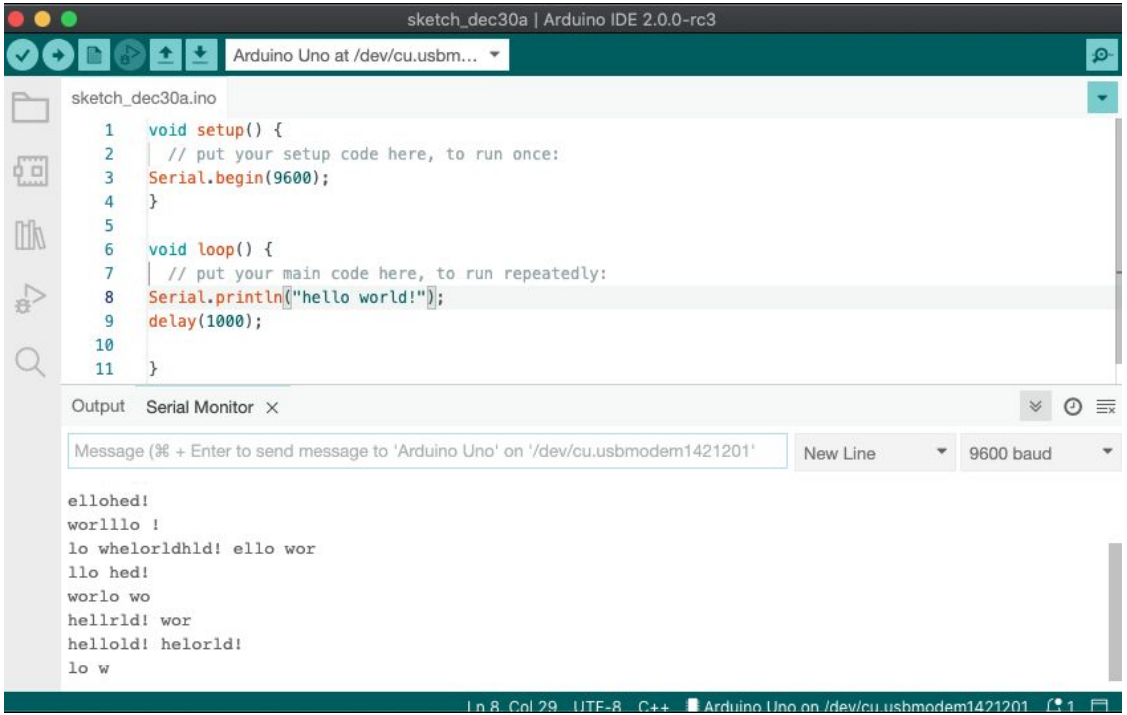
- Para facilitar o uso de componentes em circuitos com Arduino, é possível encontrar alguns módulos;
- Os módulos consistem em pequenos circuitos para facilitar o interfaceamento de algum componente específico com o Arduino;
- Tomemos como exemplo o Relé:



módulos


Arduino Relay
Flame Sensor Module
Arduino joystick
Shock switch sensor
photosensitive sensor
Ultrasonic Distance Sensor
hall sensor
Male to Male Cable
Water Sensor
Push Buttom Module
Active Buzzer Module
Push Buttom Module
Passive Buzzer Module
Touch Sensor Module
Push Buttom Module
Temperature Sensor
Push Buttom Module
Infrared Receiver Module
Push Buttom Module
Tracking Sensor Module
Temperature &
Relative Humidity Sensor Module

Ambiente de Desenvolvimento Integrado de Software para a Plataforma Arduino

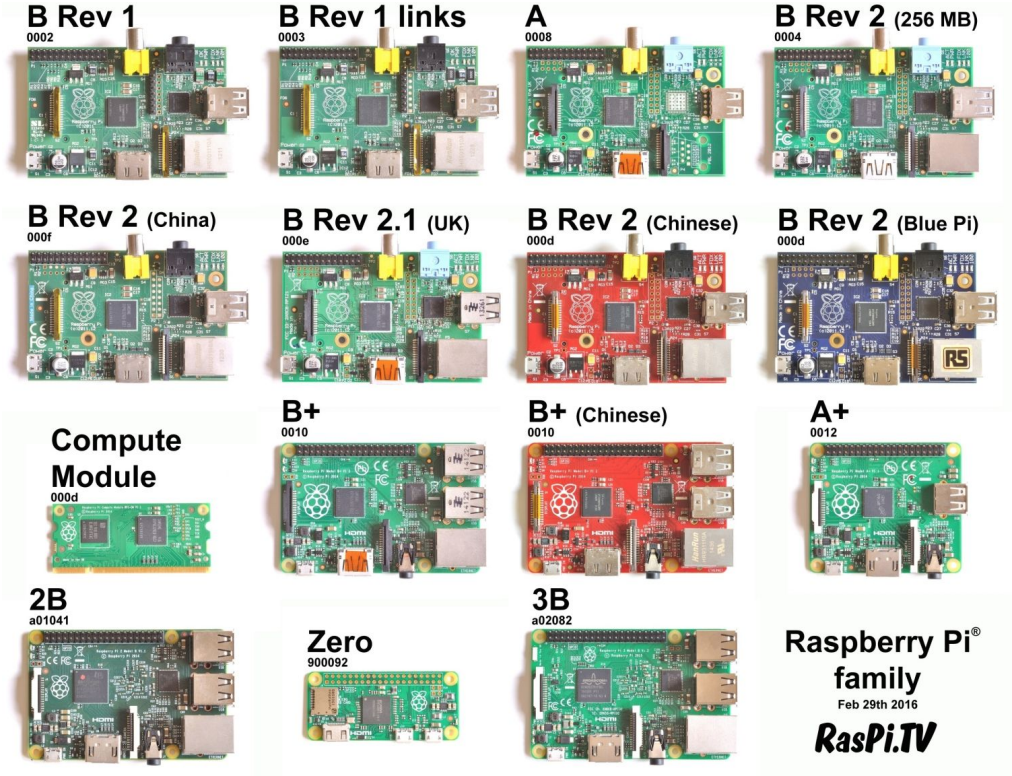


disponível em arduino.cc/en/software

RaspberryPi

- Computador do tamanho de um cartão de crédito, que se conecta a um monitor de computador ou TV, e usa um teclado e um mouse padrão
- Desenvolvido no Reino Unido pela Fundação Raspberry Pi.
- Todo o hardware é integrado numa única placa
- O principal objetivo é promover o ensino em Ciência da Computação básica em escolas

RaspberryPi




It’s a mess.

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Internet das Coisas Aplicada a Agropecuária de Precisão

Linguagem C - Introdução

FACEPE Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

SEMINE AGRICULTURA IRRIGADA INTELIGENTE

SMART RURAL

Juá Labs

histórico

- criada no Bell Laboratories por Ken Thompson entre 1969 e 1973
- linguagem compilada puramente procedural
- parte do desenvolvimento do SO Unix
- sistemas operacionais modernos como Linux, Solaris e BSD são escritos principalmente em linguagem C
- ANSI C refere-se ao padrão proposto para a linguagem C em meados dos anos 80.

```
/*********************************************
 * IoT Aplicada a Agropecuaria de Precisao
 * Exemplo 1
 *********************************************/
#include <Adafruit_NeoPixel.h>

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(24, 2, NEO_GRB + NEO_KHZ800);

uint8_t redColor = 255;
uint8_t greenColor = 0;
uint8_t blueColor = 0;

void setup() {
      // inicializa a biblioteca NeoPixel
      pixels.begin();
}

void loop() {
    for(int i= 0; i < 24; i++) {
        // pixels.Color recebe valores RGB, de 0,0,0 ate 255,255,255
        pixels.setPixelColor(i, pixels.Color(redColor, greenColor, blueColor));
        // atualiza o valor do pixel no hardware
        pixels.show();
        // delay em milisegundos
        delay(100);
      }
}
```

Links do Exemplo 1:

GitHUB

Projeto TinkerCad

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

```
/*********************************************
 * IoT Aplicada a Agropecuaria de Precisao
 * Exemplo 1
 *********************************************/
#include <Adafruit_NeoPixel.h>

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(24, 2, NEO_GRB + NEO_KHZ800);

uint8_t redColor = 255;
uint8_t greenColor = 0;
uint8_t blueColor = 0;

void setup() {
      // inicializa a biblioteca NeoPixel
      pixels.begin();
}

void loop() {
    for(int i= 0; i < 24; i++) {
        // pixels.Color recebe valores RGB, de 0,0,0 ate 255,255,255
        pixels.setPixelColor(i, pixels.Color(redColor, greenColor, blueColor));
        // atualiza o valor do pixel no hardware
        pixels.show();
        // delay em milisegundos
        delay(100);
      }
}
```

/* e */ são utilizados para comentários em blocos

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

```
/*********************************************
 * IoT Aplicada a Agropecuaria de Precisao
 * Exemplo 1
 *********************************************/
#include <Adafruit_NeoPixel.h>

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(24, 2, NEO_GRB + NEO_KHZ800);

uint8_t redColor = 255;
uint8_t greenColor = 0;
uint8_t blueColor = 0;

void setup() {
      // inicializa a biblioteca NeoPixel
      pixels.begin();
}

void loop() {
    for(int i= 0; i < 24; i++) {
        // pixels.Color recebe valores RGB, de 0,0,0 ate 255,255,255
        pixels.setPixelColor(i, pixels.Color(redColor, greenColor, blueColor));
        // atualiza o valor do pixel no hardware
        pixels.show();
        // delay em milisegundos
        delay(100);
      }
}
```

// é utilizado em comentários de linha

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

```
/*********************************************
 * IoT Aplicada a Agropecuaria de Precisao
 * Exemplo 1
 *********************************************/
#include <Adafruit_NeoPixel.h>

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(24, 2, NEO_GRB + NEO_KHZ800);

uint8_t redColor = 255;
uint8_t greenColor = 0;
uint8_t blueColor = 0;

void setup() {
      // inicializa a biblioteca NeoPixel
      pixels.begin();
}

void loop() {
    for(int i= 0; i < 24; i++) {
        // pixels.Color recebe valores RGB, de 0,0,0 ate 255,255,255
        pixels.setPixelColor(i, pixels.Color(redColor, greenColor, blueColor));
        // atualiza o valor do pixel no hardware
        pixels.show();
        // delay em milisegundos
        delay(100);
      }
}
```

{ e } abrem e fecham os blocos de código

```
/*********************************************
 * IoT Aplicada a Agropecuaria de Precisao
 * Exemplo 1
 *********************************/
#include <Adafruit_NeoPixel.h>

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(24, 2, NEO_GRB + NEO_KHZ800);

uint8_t redColor = 255;
uint8_t greenColor = 0;
uint8_t blueColor = 0;

void setup() {
      // inicializa a biblioteca NeoPixel
      pixels.begin();
}

void loop() {
    for(int i= 0; i < 24; i++) {
        // pixels.Color recebe valores RGB, de 0,0,0 ate 255,255,255
        pixels.setPixelColor(i, pixels.Color(redColor, greenColor, blueColor));
        // atualiza o valor do pixel no hardware
        pixels.show();
        // delay em milisegundos
        delay(100);
      }
}
```

#include serve para incluir bibliotecas externas

```
/*********************************************
 * IoT Aplicada a Agropecuaria de Precisao
 * Exemplo 1
 *********************************************/
#include <Adafruit_NeoPixel.h>

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(24, 2, NEO_GRB + NEO_KHZ800);

uint8_t redColor = 255;
uint8_t greenColor = 0;
uint8_t blueColor = 0;

void setup() {
      // inicializa a biblioteca NeoPixel
      pixels.begin();
}

void loop() {
    for(int i= 0; i < 24; i++) {
        // pixels.Color recebe valores RGB, de 0,0,0 ate 255,255,255
        pixels.setPixelColor(i, pixels.Color(redColor, greenColor, blueColor));
        // atualiza o valor do pixel no hardware
        pixels.show();
        // delay em milisegundos
        delay(100);
      }
}
```

declaração de variáveis globais.
Estas variáveis são do tipo uint8_t.

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Internet das Coisas Aplicada a Agropecuária de Precisão

Linguagem C - Tipos de Dados

FACEPE Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

SEMINE AGRICULTURA IRRIGADA INTELIGENTE

SMART RURAL

Juá Labs

tipos de dados padrão

Type	Description
`char`	A single byte (8 bits). Signed unless specified as unsigned.
`int`	An integer, which is typically the size of native integer values on the host system. Commonly found in either 16- or 32-bit sizes. Signed unless specified as unsigned.
`float`	A single-precision floating-point value, typically expressed in 32 bits (4 bytes). Always signed.
`double`	A double-precision floating-point value, typically expressed in 64 bits (8 bytes). Always signed.

tipos de dados estendidos e tipos definidos pelo usuário

Type	Bytes	Range
unsigned char	1	0 to 255
signed char	1	–128 to 127
char (same as signed char)	1	–128 to 127
short (int)	2	–32,768 to 32,767
unsigned short (int)	2	0 to 65,535
unsigned int	4	0 to 4,294,967,295
signed int	4	–2,147,483,648 to 2,147,483,647
int (same as signed int)	4	–2,147,483,648 to 2,147,483,647
unsigned long	4	0 to 4,294,967,295
long (int)	4	–2,147,483,648 to 2,147,483,647

```
typedef signed short    int16_t;
typedef unsigned short  uint16_t;
typedef float           float32_t;
typedef double          float64_t;
```

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Internet das Coisas Aplicada a Agropecuária de Precisão

Linguagem C - Estruturas de Controle

FACEPE Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

SEMINE AGRICULTURA IRRIGADA INTELIGENTE

SMART RURAL

Juá Labs

```
/*********************************************
 * IoT Aplicada a Agropecuaria de Precisao
 * Exemplo 1
 *********************************************/
#include <Adafruit_NeoPixel.h>

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(24, 2, NEO_GRB + NEO_KHZ800);

uint8_t redColor = 255;
uint8_t greenColor = 0;
uint8_t blueColor = 0;

void setup() {
      // inicializa a biblioteca NeoPixel
      pixels.begin();
}

void loop() {
    for(int i= 0; i < 24; i++) {
        // pixels.Color recebe valor
        pixels.setPixelColor(i, pixels.Color(redColor, greenColor, blueColor));
        // atualiza o valor do pixel no hardware
        pixels.show();
        // delay em milisegundos
        delay(100);
      }
}
```

a estrutura de controle 'for'

estruturas de controle

for

```
for(<inicializacao>; <condicao de parada>; <incremento>) {
    // comandos
}

// exemplo que imprime "Hello World!" 10 vezes na interface serial
for(int i = 0; i < 10; i++) {
    Serial.println("Hello World!");
}
```

estruturas de controle

if-else

```
if(<expressao 1>) {
    // comandos
}
else if (<expressao 2>) {
    // comandos
}
else if (<expressao 3>) {
    // comandos
}
else {
    // comandos
}
```

```
// exemplo:
int8_t y = 0;

if (x > 10) {
    y = 1;
}
else if ( x < 0 ) {
    y = -1;
}
else {
    y = 0
}
```

estruturas de controle

while

```
while(<condicao>) {
    // comandos
}

// exemplo que imprime a mensagem "O valor de 'x' e negativo" enquanto
o valor de x for menor que 0
int8_t x = -10;
while(x < 0) {
    Serial.println("O valor de 'x' e negativo");
    x++;
}
```

estruturas de controle

switch

```
switch(<expressao>) {
    case valor: // comandos
    case valor: // comandos
    case valor: // comandos
    default:    // comandos
}
```

```
switch(x) {
    case 0:
        Serial.println("x tem o valor 0");
        break;
    case 1:
        Serial.println("x tem o valor 1");
        break;
    case 2:
        Serial.println("x tem o valor 2");
        break;
    default:
        Serial.println("x tem o valor diferente de 0,
1 e 2");
}
```

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Internet das Coisas Aplicada a Agropecuária de Precisão

Linguagem C - Diretivas de compilação

FACEPE Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

SEMINE AGRICULTURA IRRIGADA INTELIGENTE

SMART RURAL

Juá Labs

```
/*********************************************
 * IoT Aplicada a Agropecuaria de Precisao
 * Exemplo 2
 *********************************************/

#include <Adafruit_NeoPixel.h>
#define PIN         2    // pino de dados do NeoPixel
#define NUMPIXELS     24     // quantidade de pixels na fita
#define DELAY        100 // tempo de delay entre pixels

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

uint8_t redColor[NUMPIXELS] = {255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0};
uint8_t greenColor[NUMPIXELS] = {0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0};
uint8_t blueColor[NUMPIXELS] =  {0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255};

void setup() {
    // inicializa a biblioteca NeoPixel
    pixels.begin();
}

void loop() {
    for(int i= 0; i < NUMPIXELS; i++) {
        // pixels.Color recebe valores RGB, de 0,0,0 ate 255,255,255
        pixels.setPixelColor(i, pixels.Color(redColor[i], greenColor[i], blueColor[i]));
        // atualiza o valor do pixel no hardware
        pixels.show();
        // delay em milisegundos
        delay(DELAY);
      }
}
```

diretivas de pré compilação

Links do Exemplo 2:

[GitHUB]

[Projeto TinkerCad]

```
/*********************************************
 * IoT Aplicada a Agropecuaria de Precisao
 * Exemplo 2
 *********************************************/

#include <Adafruit_NeoPixel.h>
#define PIN         2    // pino de dados do NeoPixel
#define NUMPIXELS     24     // quantidade de pixels na fita
#define DELAY        100 // tempo de delay entre pixels
```

diretivas de pré compilação

diretivas do pré-processador

- #include, #define, #if, #elif, #ifdef, #ifndef, #else, #endif
- #include - é utilizado para incluir os conteúdos de um arquivo em um outro
  - #include <stdio.h>
- #define - associa um nome (string) a um outro nome. usado para substituir constantes que aparecem com frequência no código.
  - #define UP 1
  - #define DOWN 0

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Internet das Coisas Aplicada a Agropecuária de Precisão

Linguagem C - Arrays

FACEPE Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

SEMINE AGRICULTURA IRRIGADA INTELIGENTE

SMART RURAL

Juá Labs

```
/*********************************************
 * IoT Aplicada a Agropecuaria de Precisao
 * Exemplo 2
 *********************************************/

#include <Adafruit_NeoPixel.h>
#define PIN         2    // pino de dados do NeoPixel
#define NUMPIXELS     24     // quantidade de pixels na fita
#define DELAY        100 // tempo de delay entre pixels

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

uint8_t redColor[NUMPIXELS] = {255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0};
uint8_t greenColor[NUMPIXELS] = {0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0};
uint8_t blueColor[NUMPIXELS] =  {0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255};

void setup() {
    // inicializa a biblioteca NeoPixel
    pixels.begin();
}

void loop() {
    for(int i= 0; i < NUMPIXELS; i++) {
        // pixels.Color recebe valores RGB, de 0,0,0 ate 255,255,255
        pixels.setPixelColor(i, pixels.Color(redColor[i], greenColor[i], blueColor[i]));
        // atualiza o valor do pixel no hardware
        pixels.show();
        // delay em milisegundos
        delay(DELAY);
      }
}
```

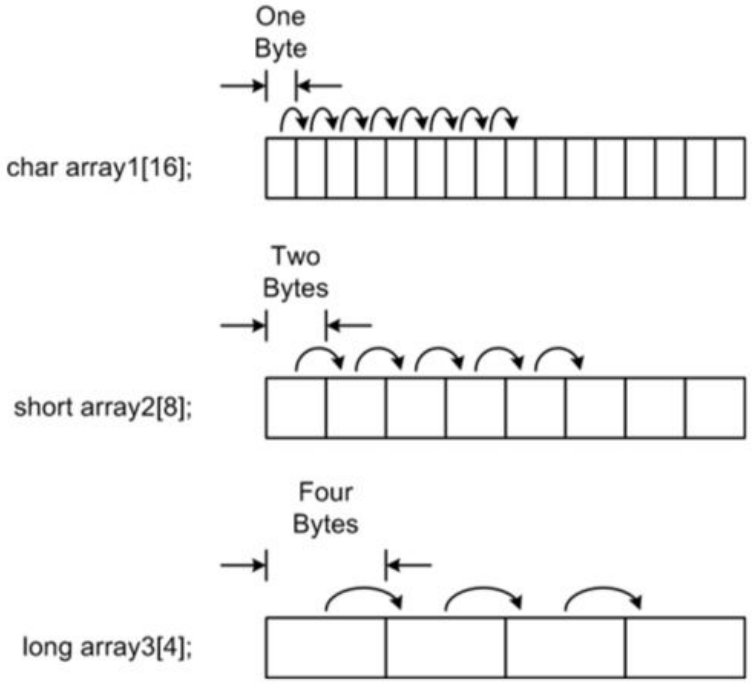
declaração de arrays

```
/*********************************************
 * IoT Aplicada a Agropecuaria de Precisao
 * Exemplo 2
 *********************************************/

#include <Adafruit_NeoPixel.h>
#define PIN         2    // pino de dados do NeoPixel
#define NUMPIXELS     24     // quantidade de pixels na fita
#define DELAY        100 // tempo de delay entre pixels

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

uint8_t redColor[NUMPIXELS] = {255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0};
uint8_t greenColor[NUMPIXELS] = {0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0};
uint8_t blueColor[NUMPIXELS] =  {0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255};

void setup() {
}

voi
}
```

declaração de arrays


```
uint8_t redColor[NUMPIXELS] = {255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0};
uint8_t greenColor[NUMPIXELS] = {0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0};
uint8_t blueColor[NUMPIXELS] =  {0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255};
```

arrays


One
Byte
char array1[16];
Two
Bytes
short array2[8];
Four
Bytes
long array3[4];

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Internet das Coisas Aplicada a Agropecuária de Precisão

Linguagem C - Estruturas (*Structs*)

FACEPE Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

SEMINE AGRICULTURA IRRIGADA INTELIGENTE

SMART RURAL

Juá Labs

```
/*********************************************
 * IoT Aplicada a Agropecuaria de Precisao
 * Exemplo 3
 *********************************************/

#include <Adafruit_NeoPixel.h>

#define PIN                 2    // pino de dados do NeoPixel
#define NUMPIXELS             24     // quantidade de pixels na fita
#define DELAY                100 // tempo de delay entre pixels
#define RAINBOW_NUM_COLORS     7

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

typedef struct RGBLED_t {
    uint8_t R;
    uint8_t G;
    uint8_t B;
} RGBLED;

const RGBLED VIOLET = {148, 0, 211};
const RGBLED INDIGO = {75, 0, 130};
const RGBLED BLUE   = {0, 0, 255};
const RGBLED GREEN  = {0, 255, 0};
const RGBLED YELLOW = {255, 255, 0};
const RGBLED ORANGE = {255, 127, 0};
const RGBLED RED    = {255, 0, 0};

RGBLED rainbow_colors[RAINBOW_NUM_COLORS] = {VIOLET, INDIGO, BLUE, GREEN, YELLOW, ORANGE, RED};

void setup() {
        // inicializa a biblioteca NeoPixel
      pixels.begin();
}
… continua no próximo slide
```

definição de estruturas

Links do Exemplo 3:

- GitHUB
- Projeto TinkerCad

```
/*********************************************
 * IoT Aplicada a Agropecuaria de Precisao
 * Exemplo 3
 *********************************************/

#include <Adafruit_NeoPixel.h>

#define PIN
#define NU
#define DE
#define RA

Adafruit_N

typedef st
    uint8_
    uint8_
    uint8_
} RGBLED;

const RGBL
const RGBL
const RGBL
const RGBL
const RGBL
const RGBL
const RGBL

RGBLED rai

void setup
        //
      pixe
}
… continua
```

```
typedef struct RGBLED_t {
    uint8_t R;
    uint8_t G;
    uint8_t B;
} RGBLED;

const RGBLED VIOLET = {148, 0, 211};
const RGBLED INDIGO = {75, 0, 130};
const RGBLED BLUE   = {0, 0, 255};
const RGBLED GREEN  = {0, 255, 0};
const RGBLED YELLOW = {255, 255, 0};
const RGBLED ORANGE = {255, 127, 0};
const RGBLED RED    = {255, 0, 0};
```

definição de estruturas

estruturas

- é uma coleção de variáveis de diferentes tipos;
- estruturas podem conter subestruturas dentro delas;

```
struct <nome_da_estrutura> {
    <tipo> <nome_da_variavel>;
    <tipo> <nome_da_variavel>;
    ...
    <tipo> <nome_da_variavel>;
};
ou
} <nome_da_instancia_da_estrutura>;
```

```
// exemplo
struct RGBLED_t {
    uint8_t R;
    uint8_t G;
    uint8_t B;
} RGBLED;

// acessando elementos da
estrutura
uint8_t green = RGBLED.G;
```

podemos definir uma estrutura como um tipo.
obs: qualquer tipo pode ser atribuído a um novo tipo.
e.g. typedef int int16_t

```
typedef struct RGBLED_t {
    uint8_t R;
    uint8_t G;
    uint8_t B;
} RGBLED;

const RGBLED VIOLET = {148, 0, 211};
const RGBLED INDIGO = {75, 0, 130};
const RGBLED BLUE   = {0, 0, 255};
const RGBLED GREEN  = {0, 255, 0};
const RGBLED YELLOW = {255, 255, 0};
const RGBLED ORANGE = {255, 127, 0};
const RGBLED RED    = {255, 0, 0};
```

```
/*********************************************
 * IoT Aplicada a Agropecuaria de Precisao
 * Exemplo 3
 *********************************************/

#include <Adafruit_NeoPixel.h>

#de
#de
#de
#de

Ada

typ

} RG

cons
cons
cons
cons
cons
cons
cons

RGBL

voi

}
… co
```

```
typedef struct RGBLED_t {
    uint8_t R;
    uint8_t G;
    uint8_t B;
} RGBLED;

const RGBLED VIOLET = {148, 0, 211};
const RGBLED INDIGO = {75, 0, 130};
const RGBLED BLUE   = {0, 0, 255};
const RGBLED GREEN  = {0, 255, 0};
const RGBLED YELLOW = {255, 255, 0};
const RGBLED ORANGE = {255, 127, 0};
const RGBLED RED    = {255, 0, 0};
```

uma vez que temos um tipo definido, podemos utilizá-lo para declarar novas variáveis ou até mesmo constantes.

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Internet das Coisas Aplicada a Agropecuária de Precisão

Linguagem C - Operadores e Expressões

FACEPE Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

SEMINE AGRICULTURA IRRIGADA INTELIGENTE

SMART RURAL

Juá Labs

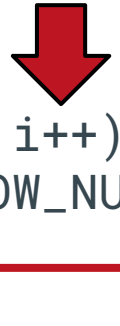
```
… continuacao

void loop() {
    uint8_t current_color = 0;
    for(int i = 0; i < NUMPIXELS; i++) {
        current_color = i % RAINBOW_NUM_COLORS;
        // pixels.Color recebe valores RGB, de 0,0,0 ate 255,255,255
        pixels.setPixelColor(i, pixels.Color(rainbow_colors[current_color].R, rainbow_colors[current_color].G,
rainbow_colors[current_color].B));
        // atualiza o valor do pixel
        pixels.show();
        // delay em milisegundos
        delay(DELAY);
      }
}
```

as linguagens de programação oferecem diversos operadores que nos permitem realizar operações lógico aritméticas sob as variáveis do nosso programa.

… continuacao

```
void loop() {
    uint8_t current_color = 0;
```

```
...
void loop() {
    uint8_t current_color = 0;
    for(int i = 0; i < NUMPIXELS; i++) {
        current_color = i % RAINBOW_NUM_COLORS;
...
```



```
rain                                                    urrent_color].G,
}
```

operadores aritméticos

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus

operadores de atribuição e operadores unários

Operator	Description
=	Basic assignment
+=	Addition *and* assignment
-=	Subtraction *and* assignment
*=	Multiplication *and* assignment
/=	Division *and* assignment
%=	Modulus *and* assignment
<<=	Assignment by bitwise left shift
>>=	Assignment by bitwise right shift
&=	Assignment by bitwise AND
|=	Assignment by bitwise OR
^=	Assignment by bitwise XOR

Operator	Description
+a	Unary plus
-a	Unary minus (arithmetic negation)
++a	Pre-increment
a++	Post-increment
--a	Pre-decrement
a--	Post-decrement
!a	Logical negation (NOT)

Operator	Description
~a	Bitwise one's complement (NOT)
*	Pointer dereference (see section on pointers)
&	Memory address (see section on pointers)

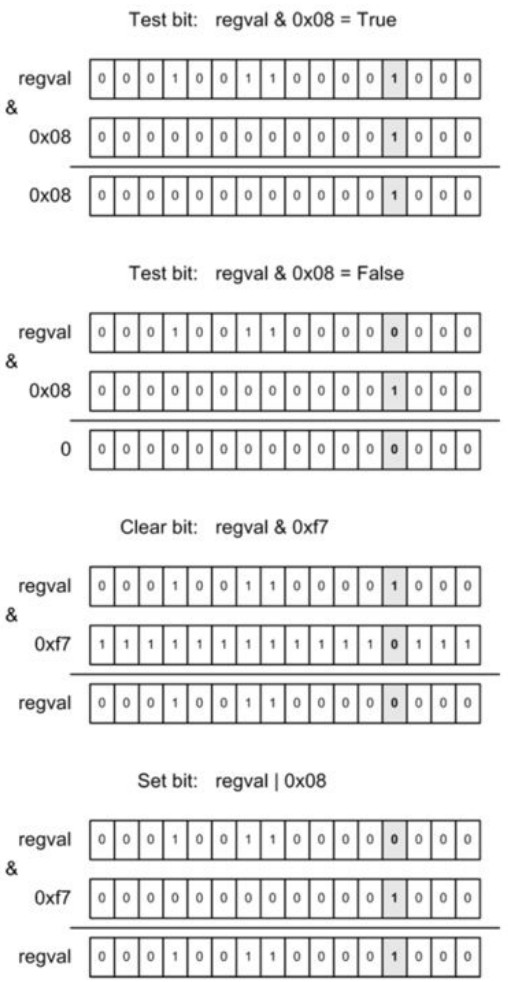
operadores lógicos e de comparação

Operator	Description
!a	Logical negation (NOT)
&&	Logical AND
||	Logical OR

Operator	Description
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
!=	Not equal to
==	Equal to

operadores *bitwise*

Operator	Description
<<	Bitwise left shift
>>	Bitwise right shift
~a	Bitwise one's complement (NOT)
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<<=	Assignment by bitwise left shift
>>=	Assignment by bitwise right shift
&=	Assignment by bitwise AND
|=	Assignment by bitwise OR
^=	Assignment by bitwise XOR



precedência dos operadores

Operator	Description	Associativity
()	Parentheses	Left to right
[]	Brackets (array subscript)	
.	Member selection via object name	
->	Member selection via pointer	
++ --	Postfix increment/decrement	
++ --	Prefix increment/decrement	Right to left
+ -	Unary plus/minus	
! ~	Logical negation/bitwise complement	
(type)	Cast (change type)	
*	Dereference	
&	Address	
sizeof	Determine size in bytes	
* / %	Multiplication/division/modulus	Left to right
+ -	Addition/subtraction	Left to right
<< >>	Bitwise shift left, bitwise shift right	Left to right
< <=	Relational less than/less than or equal to	Left to right
> >=	Relational greater than/greater than or equal to	
== !=	Relational is equal to/is not equal to	Left to right
&	Bitwise AND	Right to left
^	Bitwise exclusive OR (XOR)	Right to left
|	Bitwise inclusive OR	Right to left
&&	Logical AND	Left to right
||	Logical OR	Left to right
?:	Ternary conditional	Right to left
=	Assignment	Right to left
+= -=	Addition/subtraction assignment	
*= /=	Multiplication/division assignment	
%= &=	Modulus/bitwise AND assignment	
^= |=	Bitwise exclusive/inclusive OR assignment	
<<= >>=	Bitwise left/right shift assignment	
,	Comma (separate expressions)	Left to right

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Internet das Coisas Aplicada a Agropecuária de Precisão

Linguagem C - Ponteiros

FACEPE Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

SEMINE AGRICULTURA IRRIGADA INTELIGENTE

SMART RURAL

Juá Labs

ponteiros

- ponteiros são variáveis que contêm um endereço de memória.
- o endereço pode ser referendo a uma única variável ou a uma região contígua da memória.

```
int x = 5; // declaracao da variável inteira 'x'
int y;       // declaracao da variável inteira 'y'
int *p;  // declaracao de um ponteiro para inteiro

p = &x;  // endereco da variavel 'x' passado para
             // o ponteiro 'p'
y = *p;  // conteudo da posicao de memoria apontada
             // por 'p' atribuida a variavel 'y'
```

ponteiros em funções

- ponteiros podem ser utilizados como parâmetros de funções
- permite que dados fora das funções sejam manipulados

```
void swap (int *x, int *y) {
    int tmp;
    tmp = *y;
    *y = *x;
    *x = tmp;
}
```

```
int a = 10;
int b = 20;

swap(&a, &b);
```

ponteiros e arrays

- ponteiros e arrays em 'C' estão relacionados
- o primeiro elemento de um array corresponde ao endereço base do espaço de memória ocupado por ele

```
int anarray[10];
int *p;
p = &anarray[0];
```

```
narray[5];
*(p+5);
```

```
p = anarray;
```

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Internet das Coisas Aplicada a Agropecuária de Precisão

Linguagem C - Funções

FACEPE Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

SEMINE AGRICULTURA IRRIGADA INTELIGENTE

SMART RURAL

Juá Labs

funções

```
<tipo> <nome_da_funcao> (<parametros>) {
    // comandos
}


// exemplo
void setLEDRGB(uint8_t LED_index, RGBLED LED) {
    pixels.setPixelColor(LED_index, pixels.Color(LED.R, LED.G, LED.B));
}
```

funções

```
<tipo> <nome_da_funcao> (<parametros>) {
    // comandos
}

// exemplo
void setLEDRGB(uint8_t LED_index, RGBLED LED) {
    pixels.setPixelColor(LED_index, pixels.Color(LED.R, LED.G, LED.B));
}
```

como esta função ficaria no nosso código?

```
/***********************************************
 * IoT Aplicada a Agropecuaria de Precisao
 * Exemplo 4
 ***********************************************/
#include <Adafruit_NeoPixel.h>

#define PIN                 2    // pino de dados do NeoPixel
#define NUMPIXELS             24     // quantidade de pixels na fita
#define DELAY                100 // tempo de delay entre pixels
#define RAINBOW_NUM_COLORS     7

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

typedef struct RGBLED_t {
    uint8_t R;
    uint8_t G;
    uint8_t B;
} RGBLED;

const RGBLED VIOLET = {148, 0, 211};
const RGBLED INDIGO = {75, 0, 130};
const RGBLED BLUE   = {0, 0, 255};
const RGBLED GREEN  = {0, 255, 0};
const RGBLED YELLOW = {255, 255, 0};
const RGBLED ORANGE = {255, 127, 0};
const RGBLED RED    = {255, 0, 0};

RGBLED rainbow_colors[NUMPIXELS] = {VIOLET, INDIGO, BLUE, GREEN, YELLOW, ORANGE, RED};

void setRGBLED(uint8_t LED_index, struct RGBLED_t LED) {
    pixels.setPixelColor(LED_index, pixels.Color(LED.R, LED.G, LED.B));
}

void setup() {
      // inicializa a biblioteca NeoPixel
      pixels.begin();
}

void loop() {
    uint8_t current_color = 0;
    for(int i = 0; i < NUMPIXELS; i++) {
        current_color = i % RAINBOW_NUM_COLORS;
        // setRGBLED recebe o indice do LED que ira acender um valor RGBLED que contem as cores
        setRGBLED(i, rainbow_colors[current_color]);
        // atualiza o valor do pixel no hardware
        pixels.show();
        // delay em milisegundos
        delay(DELAY);
      }
}
```

Links do Exemplo 4:

- GitHUB
- Projeto TinkerCad

```
/***********************************************
 * IoT Aplicada a Agropecuaria de Precisao
 * Exemplo 4
 ***********************************************/
#include <Adafruit_NeoPixel.h>

#define PIN                 2    // pino de dados do NeoPixel
#define NUMPIXELS             24     // quantidade de pixels na fita
#define DELAY               100 // tempo de delay entre pixels
#define RAINBOW_NUM_COLORS     7

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

typedef struct RGBLED_t {
    uint8_t R;
    uint8_t G;
    uint8_t B;
} RGBLED;

...
void setRGBLED(uint8_t LED_index, struct RGBLED_t LED) {
    pixels.setPixelColor(LED_index, pixels.Color(LED.R, LED.G, LED.B));
}
...
void loop() {
    uint8_t current_color = 0;
    for(int i = 0; i < NUMPIXELS; i++) {
        current_color = i % RAINBOW_NUM_COLORS;
        // setRGBLED recebe o indice do LED que ira acender um valor RGBLED que contem as cores
        setRGBLED(i, rainbow_colors[current_color]);
        // atualiza o valor do pixel no hardware
...
```

declaração da função

```
/***********************************************
 * IoT Aplicada a Agropecuaria de Precisao
 * Exemplo 4
 ***********************************************/
#include <Adafruit_NeoPixel.h>

#define PIN                 2    // pino de dados do NeoPixel
#define NUMPIXELS             24     // quantidade de pixels na fita
#define DELAY                100 // tempo de delay entre pixels
#define RAINBOW_NUM_COLORS     7

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

typedef struct RGBLED_t {
    uint8_t R;
    uint8_t G;
    uint8_t B;
} RGBLED;

...
void setRGBLED(uint8_t LED_index, struct RGBLED_t LED) {
    pixels.setPixelColor(LED_index, pixels.Color(LED.R, LED.G, LED.B));
}
...
void loop() {
    uint8_t current_colo
    for(int i = 0; i < N
        current_color = i %
        // setRGBLED recebe o indice do LED que ira acender um valor RGBLED que contem as cores
        setRGBLED(i, rainbow_colors[current_color]);
        // atualiza o valor do pixel no hardware
...
}
```

chamada da função

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Internet das Coisas Aplicada a Agropecuária de Precisão

Linguagem C - Processo de Compilação

FACEPE Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco

SEMINE AGRICULTURA IRRIGADA INTELIGENTE

SMART RURAL

Juá Labs

processo de compilação