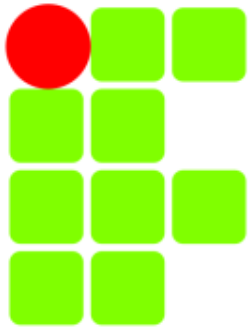


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São Carlos

Pós-graduação *Lato Sensu* em Desenvolvimento de Sistemas para Dispositivos Móveis

Android + Arduino

Tópicos em Desenvolvimento de Programas Móveis I

Alunos:

EVANDRO APARECIDO GONÇALVES
Matrícula:
SC1702581
evandroagon@gmail.com

MOISES FRANCISCO OLIMPIO FILHO
Matrícula:
SC1702742
moiza.olimpio@gmail.com

MICHEL ADRIANO MEDEIROS
Matrícula:
SC1702599
michel.medeiros@ti.rc.sp.gov.br

Professor:

PABLO ALBERTO DALBEM DE CASTRO
dalbem@ifsp.edu.br

IFSP –Campus São Carlos – 2018

Android + Arduino

Controlando Arduino via Bluetooth pelo celular

Android + Arduino

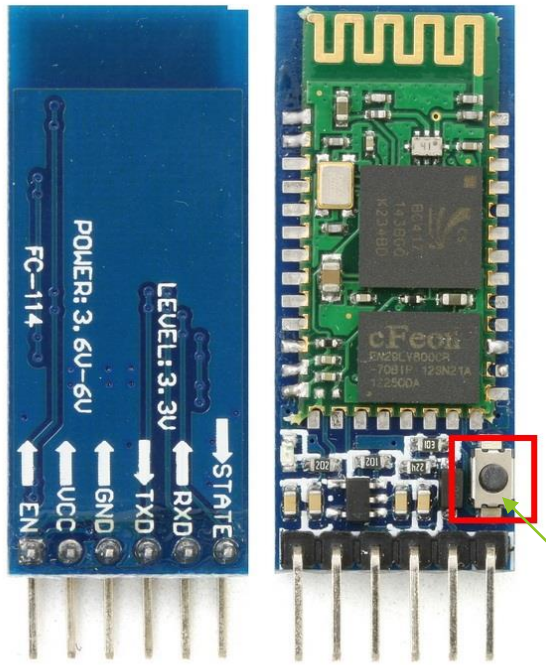
Controlando Arduino via Bluetooth pelo celular

Lista de Componentes necessários para o projeto

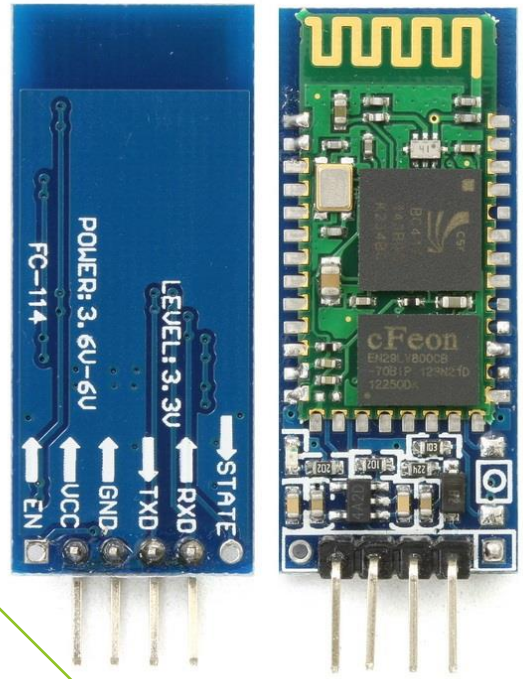
- LDR – Sensor de Luz
- Fios Jumper’s
- Protoboard
- Arduino Uno Rev3
- Módulo HC-05 ou HC-06
- LED
- Resistores 330 Ω
- Resistores de 1KΩ
- Resistores 10k Ω (funciona melhor 10k)

HC-05 ou HC-06 MODULOS BLUETOOTH

HC-05 FC-114

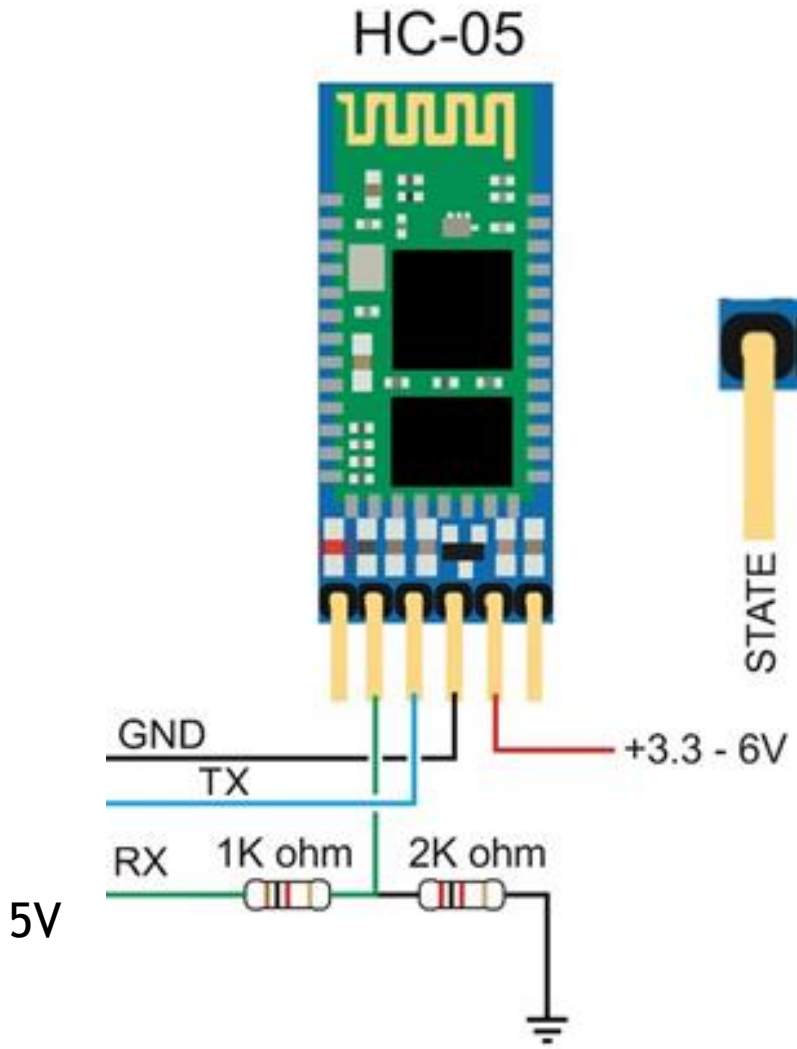


HC-06 FC-114



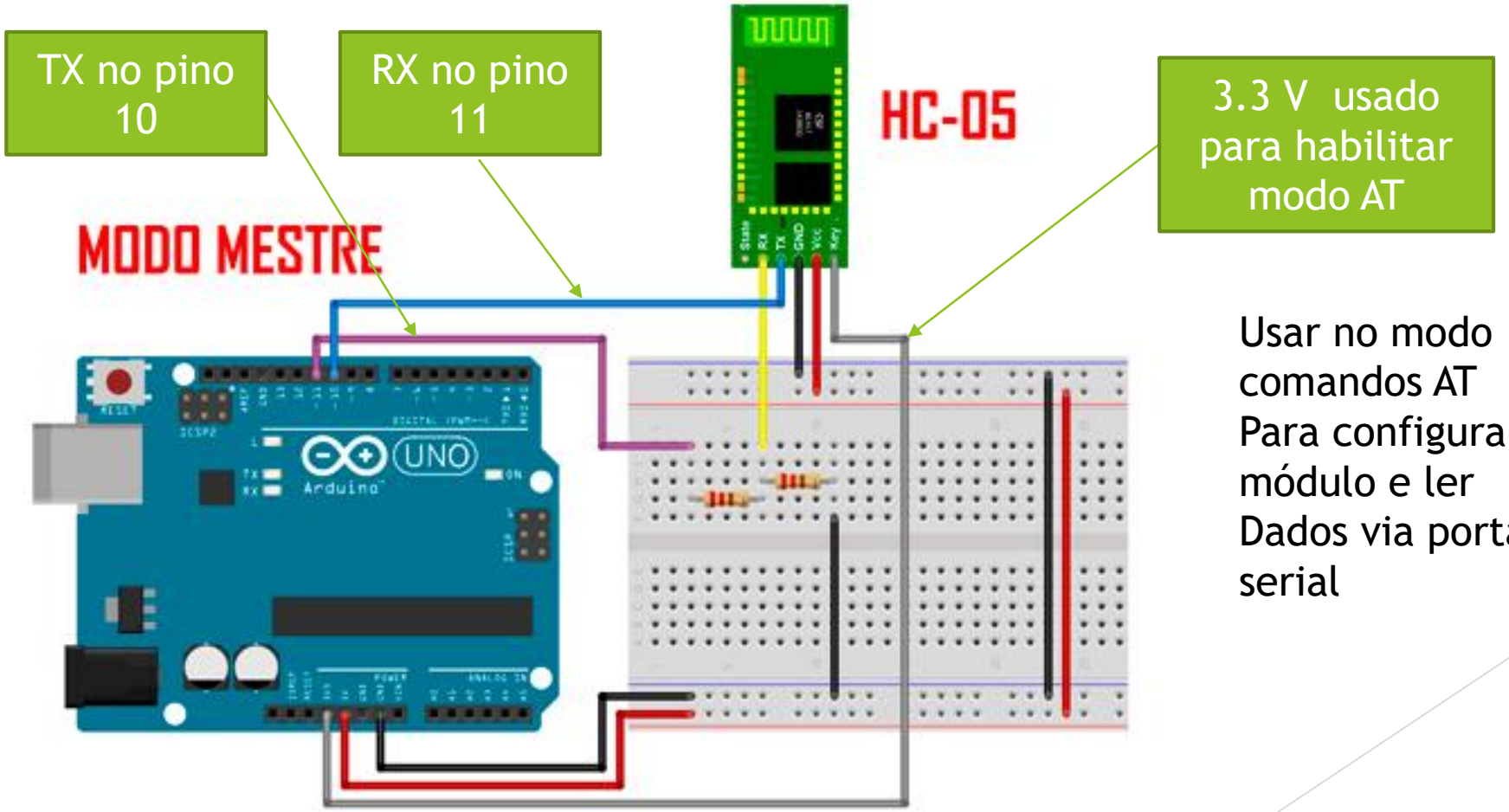
Botão para habilitar modo AT

HC-05 ou HC-06



DIVISOR DE TENSÃO para nível lógico de 3.3v

LIGAÇÃO DO HC-05 ao Arduino



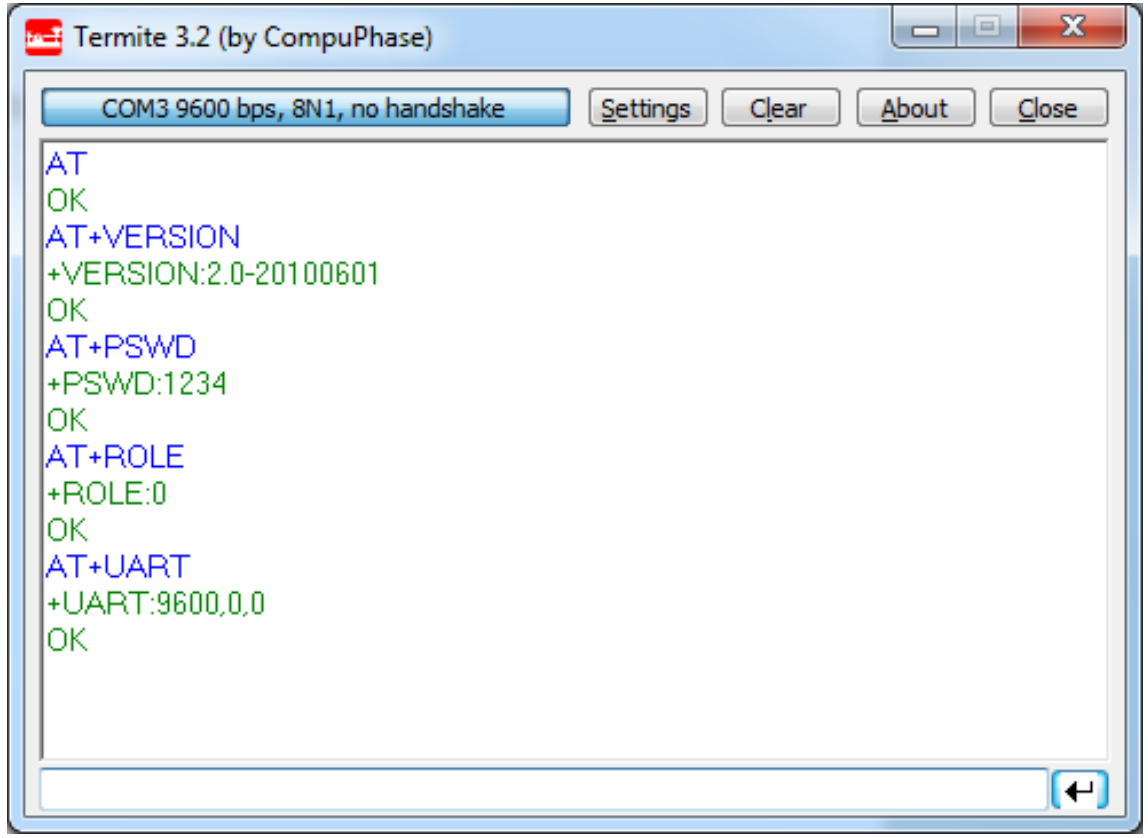
Usar no modo comandos AT
Para configurar o módulo e ler Dados via porta serial

Programa para Arduino gerenciar duas seriais (somente para configuração do H5-05)

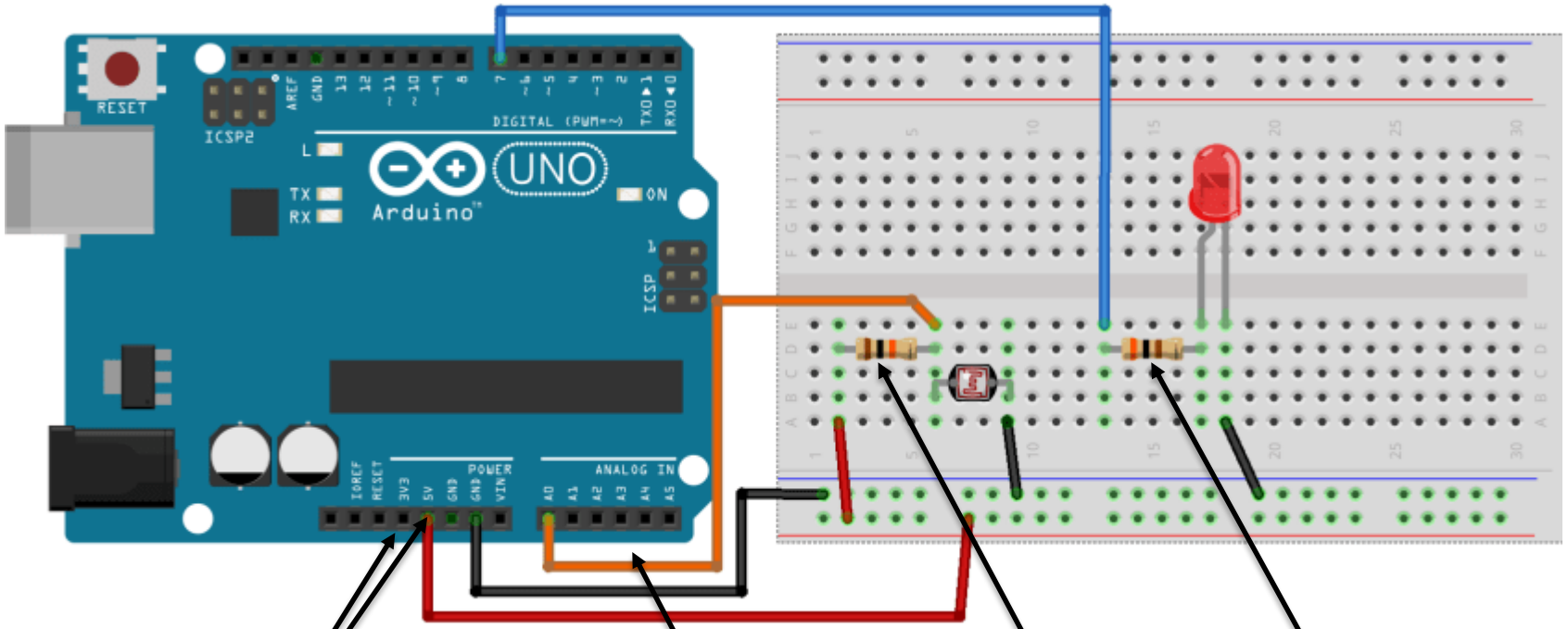
```
#include <SoftwareSerial.h>

SoftwareSerial bluetooth(10, 11); // RX, TX

void setup() {
        delay(500);
        Serial.begin(9600);  //serial do Arduino e pc
        Serial.println("Digite os comandos AT :");
        bluetooth.begin(9600);   //serial  do hc-05
}
void loop() {
        if (bluetooth.available())
        Serial.write(bluetooth.read());
        if (Serial.available())
        bluetooth.write(Serial.read());

}
```

CONFIGURANDO O MODULO BLUETOOTH

GERALMENTE OS MÓDULOS VEM CONFIGURADOS A 9600 BPS
Usar qualquer programa para comunicação via porta serial
Na IDE do Arduino já tem o Monitor Serial



Ligação do sensor de iluminação



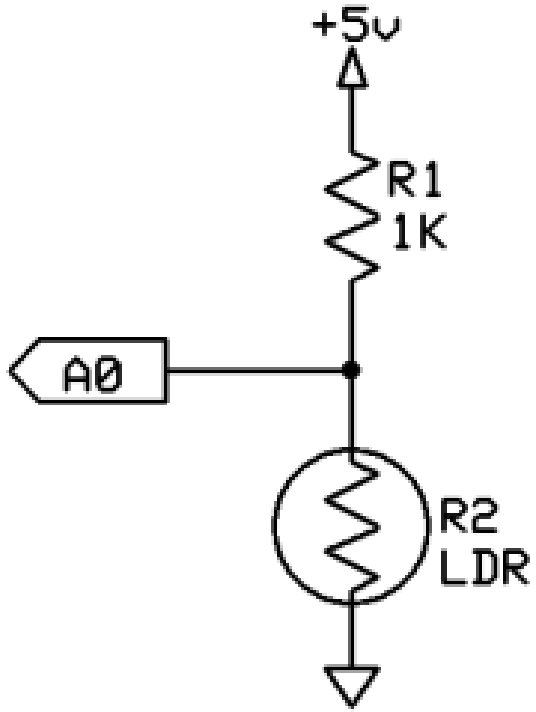
Pode ser 3v a 5v

Porta Analógica
A0 – pode ser qualquer
uma das 5

Resistor de 10kΩ

Resistor de 330Ω

Esquema Elétrico LDR



Dica! Testar o resistor ideal na prática entre 1 e 10k (depende do sensor ou ver no datasheet).

Porque a leitura analógica na prática se mostrou abaixo do esperado: O sensor estava limitando entre 0 e 255

Com 10K resistor ficou em 0 e 1024 (+ ou -)

Poderia compensar no software mas perderia sensibilidade.

Detalhes podem ser encontrados neste site:
https://portal.vidadesilicio.com.br/sensor-de-luz-com-ldr/

LENDO DADOS DO MODULO BLUETOOTH

ANTES DE USAR É NECESSÁRIO SABER SE ESTÁ CONFIGURADO CORRETAMENTE

USAR COMANDOS AT: (ESSES MÓDULOS JÁ VEM COM OPÇÃO DE TRABALHAR EM MODO AT)

USAR UM PROGRAMA PARA LER DADOS DA PORTA SERIAL

USAR UM CONVERSOR USB PARA SERIAL : DICA:

PODE SER UMA PLACA ARDUINO SEM O CHIP – NESTE CASO LIGAR TX no TX E RX no RX

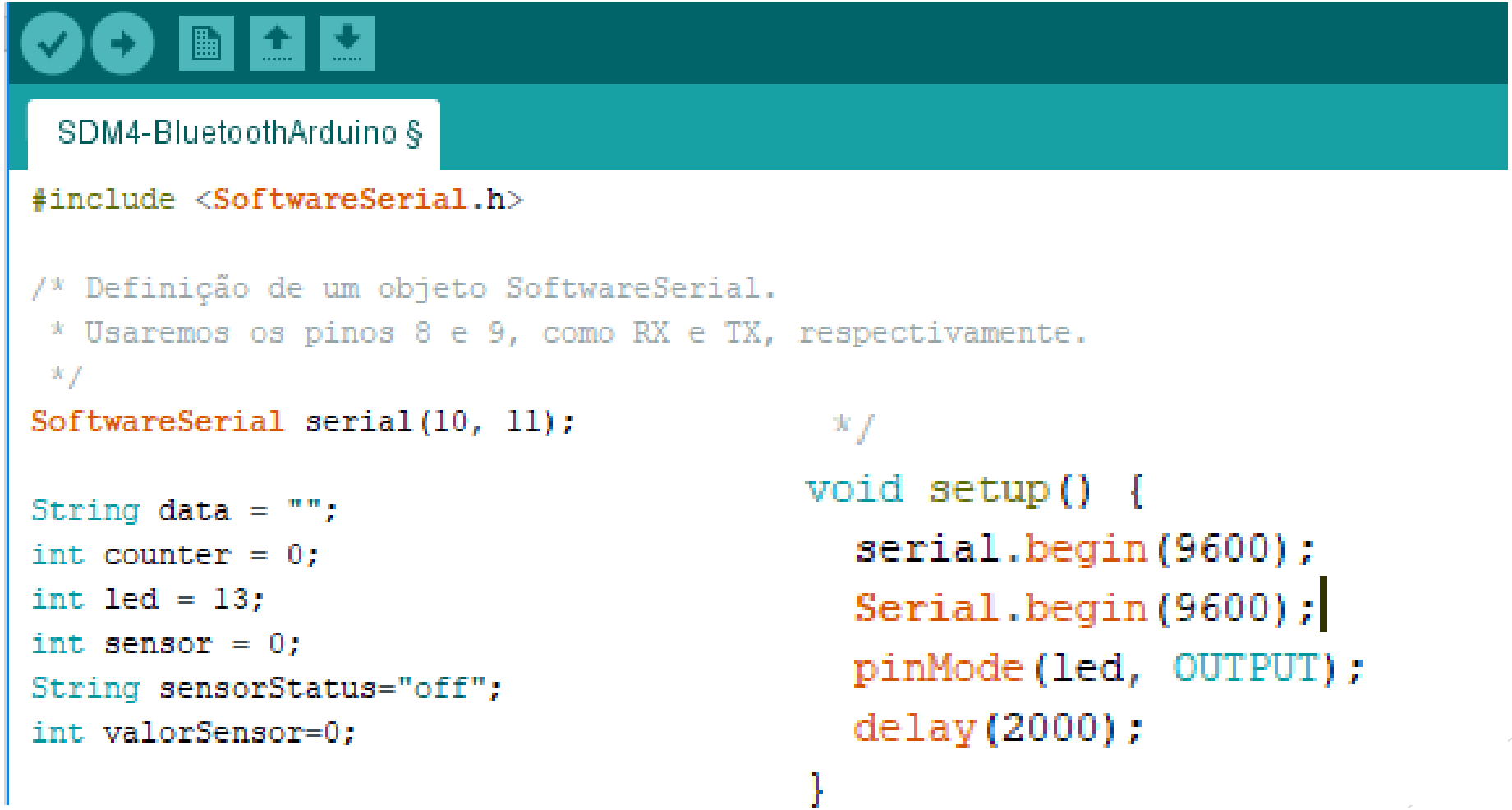
TAMBÉM PODE SER USADA A PLACA ARDUINO COM O CHIP MAS PRECISA UM PROGRAMA PARA

MUDAR A SERIAL PADRÃO PARA OUTROS PINOS PARA NÃO CONFLITAR COM A COMUNICAÇÃO ENTRE ARDUINO E O PC (USA UMA PORTA SERIAL VIRUTAL USANDO DOIS PINOS LÓGICOS E A BIBLIOTECA SOFTWARESERAL).

- MAIORES DETALHES SIGA ESTE TUTORIAL:

- https://arduinovacao.blogspot.com/2016/07/dica-comandos-at-no-modulo-bluetooth-hc.html

CODIGO DO ARDUINO



SDM4-BluetoothArduino §

```
#include <SoftwareSerial.h>

/* Definição de um objeto SoftwareSerial.
 * Usaremos os pinos 8 e 9, como RX e TX, respectivamente.
 */
SoftwareSerial serial(10, 11);

String data = "";
int counter = 0;
int led = 13;
int sensor = 0;
String sensorStatus="off";
int valorSensor=0;
```

```
 */
void setup() {
  serial.begin(9600);
  Serial.begin(9600);
  pinMode(led, OUTPUT);
  delay(2000);
}
```

CODIGO DO ARDUINO

```
void loop() {
  while(serial.available() > 0) {
    data += char(serial.read());
  }
  valorSensor=analogRead(sensor);
  if(data == "restart\n") {
    digitalWrite(led, HIGH);
    sensorStatus="off\n";
    counter = 0;
    delay(1000);
    digitalWrite(led, LOW);
    Serial.println("CONTAGEM REINICIADA");
  }
  if(data =="sensorOn\n"){
    sensorStatus="on\n";
    Serial.println("SENSOR ESTÁ EM :"+sensorStatus);
    digitalWrite(led, HIGH);
    delay(1000);
    digitalWrite(led, LOW);
  }
```

```
}
if(data == "sensorOff\n"){
  sensorStatus="off\n";
  valorSensor=0;
  digitalWrite(led, HIGH);
  delay(10);
  digitalWrite(led, LOW);
  Serial.println("SENSOR ESTÁ EM :"+sensorStatus);
}
```

CODIGO DO ARDUINO

```
  if (sensorStatus=="on\n"){
     serial.print("L: "+(String(valorSensor)+"*\n"));
     //serial.print('\n');
     Serial.println("SENSOR-----:"+String(valorSensor));
     Serial.println("**VALOR DO CONTADOR **********"+String(counter)+"*******VALOR DO CONTADOR*****************");
  }
   serial.print(String(counter));
   serial.print('\n');

  /* Finalmente, incrementamos o contador e limpamos data.
   */
  counter = counter + 1;
  data = "";
  /* Um pequeno delay para evitar bugs estranhos.
   */
  delay(10);  // SE REDUZIR O DELAY NO MEU CASO O BUFFER FICA CHEIO E NÃO LIBERA PARA NOVAS MENSAGENS
}
```

CONFIGURANDO O APP PARA O SEU HADWARE

CONFIGURANDO O UUID na Thread de conexão

```
public class ConnectionThread extends Thread{

    BluetoothSocket btSocket = null;
    BluetoothServerSocket btServerSocket = null;
    InputStream input = null;
    OutputStream output = null;
    String btDevAddress = null;
    String myUUID = "00001101-0000-1000-8000-00805F9B34FB";   // insira aqui o UUID do seu disp
    boolean server;
    boolean running = false;
    boolean isConnected = false;

    /*  Este construtor prepara o dispositivo para atuar como servidor.
     */
    public ConnectionThread() {

        this.server = true;
    }
```

INSERIR O UUID

CONFIGURANDO O APP PARA O SEU HADWARE

CONFIGURANDO O MAC DO SEU BLUETOOTH

```
 /* Definição da thread de conexão como cliente.
        Aqui, você deve incluir o endereço MAC do seu módulo Bluetooth.
        O app iniciará e vai automaticamente buscar por esse endereço.
        Caso não encontre, dirá que houve um erro de conexão.
     */
    connect = new ConnectionThread("00:21:13:03:5B:0D");  // insira o MAC do seu
dispositivo aqui
    connect.start();

    /* Um descanso rápido, para evitar bugs esquisitos.
     */
    try {
        Thread.sleep(1000);
    } catch (Exception E) {
        E.printStackTrace();
    }

}
```

INSERIR O MAC

APLICATIVO ANDROID

INSTANCIAR O ADAPTER

```
BluetoothAdapter btAdapter =
BluetoothAdapter.getDefaultAdapter();
if (btAdapter == null) {
    statusMessage.setText("Que pena! Hardware Bluetooth não está
funcionando :(");
} else {
    statusMessage.setText("Ótimo! Hardware Bluetooth está
funcionando :)");
}

});

btAdapter.enable();
```

APLICATIVO ANDROID

CLASSE ASSOCIADA AO CLICK DO BOTÃO PARA INICIAR A LEITURA DO SENSOR

```
btReiniciar.setOnClickListener(new
View.OnClickListener(){
    @Override
    public void onClick(View v) {

connect.write("sensorOn\n".getBytes());
        Toast.makeText(getBaseContext(),
"Turn on LED", Toast.LENGTH_SHORT).show();
    }
```

APLICATIVO ANDROID

HANDLER QUE TRATA AS MENSAGENS

```
public static Handler handler = handleMessage(msg) → {

        /* Esse método é invocado na Activity principal
            sempre que a thread de conexão Bluetooth recebe
            uma mensagem.
         */
        Bundle bundle = msg.getData();
        byte[] data = bundle.getByteArray( key: "data");
        String dataString= new String(data);

        /* Aqui ocorre a decisão de ação, baseada na string
            recebida. Caso a string corresponda à uma das
            mensagens de status de conexão (iniciadas com --),
            atualizamos o status da conexão conforme o código.
         */
        if(dataString.equals("---N"))
            statusMessage.setText("Ocorreu um erro durante a conexão D:");
        else if(dataString.equals("---S"))
            statusMessage.setText("Conectado :D");
        else{
            if (dataString.equals("Sensor Desligado"))
                statusMessage.setText("Conectado e sensor de luz off :D");
```

APLICATIVO ANDROID

METODO DE TRATAMENTO DAS MENSAGENS

```
public static Handler handler = new Handler() {
        @Override
        public void handleMessage(Message msg) {


 /* Esse método é invocado na Activity principal
                sempre que a thread de conexão Bluetooth
recebe
                uma mensagem.      */
```

APLICATIVO ANDROID

RECEBENDO UM BUNDLE COM A MENSAGEM

```
Bundle bundle = msg.getData();
            byte[] data = bundle.getByteArray("data");
            String dataString= new String(data);

            /* Aqui ocorre a decisão de ação, baseada na string
                recebida. Caso a string corresponda à uma das
                mensagens de status de conexão (iniciadas com --),
                atualizamos o status da conexão conforme o código.
```

APLICATIVO ANDROID

TRATANDO AS STRINGS VINDAS DA THREAD DE CONEXÃO

```
 */
if(dataString.equals("---N"))
    statusMessage.setText("Ocorreu um erro durante a conexão D:");
else if(dataString.equals("---S"))
    statusMessage.setText("Conectado :D");
else{
    if (dataString.equals("Sensor Desligado"))
        statusMessage.setText("Conectado e sensor de luz off :D");
        else{

        /* Se a mensagem não for um código de status,
        então ela deve ser tratada pelo aplicativo
        como uma mensagem vinda diretamente do outro
        lado da conexão. Nesse caso, simplesmente
        atualizamos o valor contido no TextView do
        contador.
```

APLICATIVO ANDROID

TRATANDO OS DADOS RECEBIDOS DO ARDUINO

```
if (dataString.length()>0) {
    String dataStringX = dataString.substring(0,1);
    String bufferOriginal = dataString;
    if (dataStringX.equals("L")) {
       if (bufferOriginal.length()>=4) {
           statusSensor.setText("LDR:
"+bufferOriginal.substring(3));
       }else{
              statusSensor.setText("LDR: "+bufferOriginal);
            }
        } else {
             counterMessage.setText("CONT:"+bufferOriginal);

         }
};
```

APLICATIVO ANDROID

REINICIANDO A CONTAGEM

```
    /* Esse método é invocado sempre que o usuário clicar na TextView
        que contem o contador. O app Android transmite a string "restart",
        seguido de uma quebra de linha, que é o indicador de fim de mensagem.
     */
    public void restartCounter(View view) {
        connect.write("restart\n".getBytes());
    }

}
```

VIDEO

VENDO O CONJUNTO EM FUNCIONAMENTO

Tela do Aplicativo de controle

OBS: precisa parear o celular com o módulo bluetooth antes de usar o aplicativo
Senha padrão do HC-05: 1234

Tela da interface do celular bem simples

Botão inicia no Arduino o envio da leitura do sensor LDR (sensor de Luz)

Recebe dados de um contador implementado via código do Arduino.
Ao tocar na tela um comando é enviado via bluetooth para o contador ser zerado e piscar o Led por 1 segundo mostrando que recebeu o comando

Ao receber o comando a placa do Arduino manda a leitura do sensor.

Mostra o status da conexão no rodapé da tela

Fontes

Bluetooth Android

https://developer.android.com/guide/topics/connectivity/bluetooth?hl=pt-br#ManagingAConnection

Arduino e Internet das coisas

JAVED, Adeel. Building Arduino Projects for the Internet of Things. **Experiments with Real-World Applications. United States of America: Apress Media, LLC**, 2016.

https://github.com/apress/building-arduino-projects-for-internet-of-things

http://file.allitebooks.com/20160619/Building%20Arduino%20Projects%20for%20the%20Internet%20of%20Things.pdf

https://github.com/dragaosemchama

https://github.com/thanhvie/Arduino-and-Android-communication-using-HC05

https://eletronicaparatodos.com/como-ligar-um-led-utilizando-sensor-ldr-fotoresistor-com-arduino/

Fontes

Repositório com os códigos fontes e apresentação

https://github.com/evandroagon/BluetoothArduinoAndroid.git