

27 de Junho de 2015

Primeiro dia de organização das tarefas

To Do, Doing, Done

Como funciona

Bom, antes de começar qualquer projeto gosto de planejar tudo o que tenho que fazer, pois assim consigo administrar tudo o que tenho que fazer.

Este método funciona da seguinte maneira, eu divido em três partes o desenvolvimento do projeto, em coisas que precisam ser feitas (To Do), em coisas estão sendo feitas e em coisas já terminadas.

Utilizei post-it para ter uma melhor visualização de um contexto geral do projeto e conseguir medir o que falta e o que não falta a ser feito.

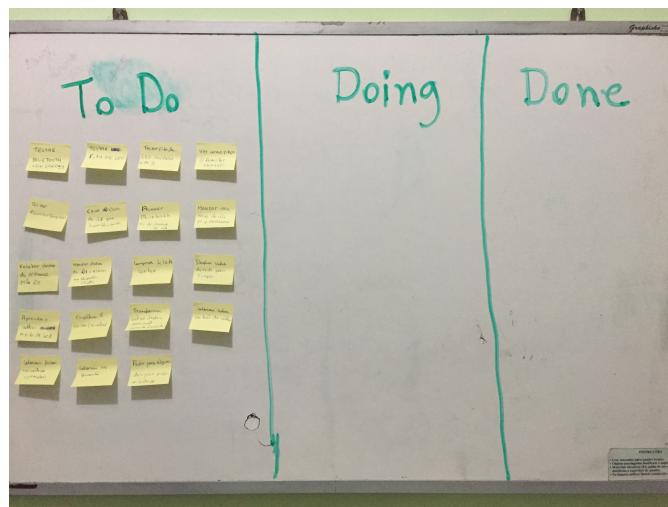


Figure 1: Lousa com Post-its

Bluetooth Low Energy 4.0

Testando Bluetooth Low Energy no Arduino

Foi feito um circuito simples que ligasse o Bluetooth Low Energy, chamado também de BLE, que fosse testado se era possível achar pelo iOS o BLE ligado ao Arduino.

Foi percebido (depois de um longo tempo tentando identificar o problema) que no menu do Bluetooth do iOS é impossível achar Bluetooth's que não são dos dispositivos da Apple. Portanto, é necessário baixar um aplicativo externo para testá-lo, no meu caso eu fiz o download do LightBlue, disponível na AppStore para iPhone.

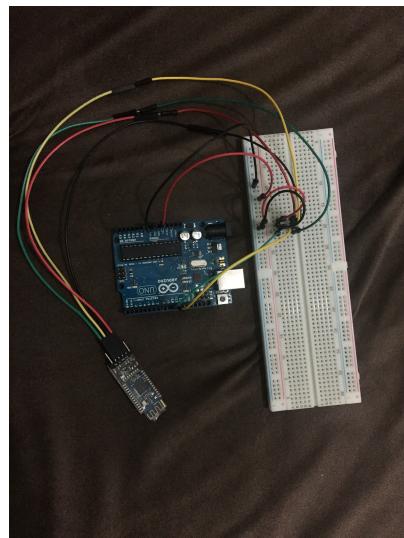


Figure 2: Teste com BLE no Arduino

28 de Junho de 2015

Segundo dia de organização das tarefas

Criando código para acessar o BLE

Em Swift

Bom, eu procurei diversos tutoriais para iOS para se conectar com o BLE 4.0, mas em Swift eu só achei um projeto chamado "BleModuleMaster" e a partir dele, eu comentei o código e futuramente pretendo criar uma biblioteca para que qualquer um possa utilizar.

O Aplicativo hoje só tem apenas um botão e uma tableView, que mostra todos os dispositivos próximos. Quando eu clico em um destes dispositivos ele conecta com o Bluetooth que eu indiquei.

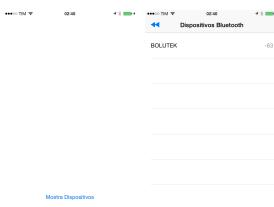


Figure 3: Primeiras imagens do Aplicativo

A parte mais complicada de hoje foi tentar enviar os dados do iOS para o Arduino, eu pensava que existia uma grande conversão entre bites e bytes e pensava que seria mais complicado que eu pensava. O algoritmo na ordem de execução fica desta forma:

- iOS - Primeiro a função recebe a String
- iOS - Depois ela converte essa String em formato de NSData(data eu digo em formato de buffers)
- iOS - Mando os dados para o Arduino
- Arduino - Lê caractere por caractere
- Arduino - Retorna pro iOS o mesmo caractere enviado.
- iOS - Receberá o caractere enviado

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(3,4); // RX, TX
char a;
void setup() {
    //Declara os seriais
    Serial.begin(9600);
    mySerial.begin(9600);

    delay(100);

    Serial.println("Setup done");
}

void loop() {
    // Verifica se recebeu algo
    if (mySerial.available()){
        Serial.print("Received data: ");

        //Enquanto tem caracteres ele continua lendo
        while (mySerial.available()) {
            a = (char) mySerial.read();

            Serial.print(a);
            if (a != NULL)
            {
                mySerial.write(a); // loop back iOS
            }
        }
        Serial.println("");
        delay(1000);
    }

    delay(1000);
}
```

```
//Escreve um valor novo no BLE
func.writeValue(data: String) {
    let data = (data as NSString).dataUsingEncoding(NSUTF8StringEncoding)
    print("data: \(data)")

    if let peripheralDevice = peripheralDevice{
        print("peripheralDevice: \(peripheralDevice)")

        if let deviceCharacteristics = deviceCharacteristics {
            print("deviceCharacteristics: \(deviceCharacteristics)")

            //Pega a String e manda para o arduino
            peripheralDevice.writeValue(data, forCharacteristic: deviceCharacteristics, type: CBCharacteristicTypeWithResponse)
            print("Escreve no arduino")
        }
    }
}
```

```
// Quando é recebido algo do Device - Retorno
func peripheral(peripheral: CPPeripheral, didUpdateValueForCharacteristic characteristic: CBCharacteristic, error: NSError!) {
    let ret = NSString(data:characteristic.value!, encoding:NSUTF8StringEncoding) as String
    print("Retorno - \(ret)")
}
```

Figure 4: Imagens do código mostrando os primeiros exemplos de leitura e captura de dados via bluetooth com Swift