Ligação ESP-8266 via I2C com ATtynni85

- Material

1-ESP-8266

1-ATtynni85

1-Level shifter

1-Ldr

1-Cristal de 16MHz

1-Resitencia 10k

4-Resistencias de 4,7k

Breadboard e cabos

- Problema

O ESP-8266 consegue comunicar por I2C, o ATtynni85 não, mas usando USI é possível arranjar maneira de ser compatível com Philips' I2C protocol. Uma vez ligados por I2C verificou-se que o clock interno do ATtynni85 não era estável tendo de ser usado um cristal externo, isto levou a outro problema pois os pinos do cristal era onde estávamos a fazer a leitura analógica, e assim tivemos de usar o pino de RESET para fazer a mesma, visto que sempre que se fazia uma leitura abaixo dos 2,2 V o ATtynni85 reiniciava, foi necessário trocar os fuses do mesmo desativando assim o pino de RESET e passando este a ser ADC0, com isto foi possível ligar ambos os dispositivos pelo protocolo pedido e enviar informações do sensor lido.

- Codigo ESP-8266 (Master)

```
#include <Wire.h>

int loopCount = 0;

 void setup(){

    Wire.begin(0,2);

    Wire.setClockStretchLimit(40000);    // in µs

    Serial.begin(115200);
```

```
     Serial.println("I'm in setup");
}
 void loop(){
  loopCount++;
  Serial.print("I'm in loop number ");
  Serial.println(loopCount);
    String dataString;
    for(int address = 1; address < 127; address++ ){
      // The i2c_scanner uses the return value of
      // the Write.endTransmisstion to see if
      // a device did acknowledge to the address.
      Wire.beginTransmission(address);
      delay(1);
      if (Wire.endTransmission() == 0){
        Serial.print("Found device with address: ");
        Serial.println(address);
        delay(1);
        Wire.requestFrom(address, 1);
        byte x = Wire.read();
        int numOfSensors = int(x);
        Serial.print("I have ");
        Serial.print(numOfSensors);
        Serial.println(" sensors");
         while (numOfSensors>0) {
          Wire.requestFrom(address, 1);
          byte y = Wire.read();
          int charsToRead = int(y);
          Serial.print("I have ");
          Serial.print(charsToRead);
          Serial.println(" chars to read");
            Wire.requestFrom(address, charsToRead);
            while (charsToRead>0) { // slave may send less than requested
              char z = Wire.read(); // receive a byte as character
              Serial.print(z);         // print the character
              dataString += z;
      charsToRead--;
     }
     numOfSensors--;
   }
  }
 }
 Serial.println();
  Serial.print("Json: ");
  Serial.println(dataString);
  Serial.println();
 delay(2000);
}
```

- Codigo ATtynni85

```
#include <TinyWireS.h>
#include <stdlib.h>
#define  HFUSE  0x5F   // Defaults for ATtiny25/45/85
#define  LFUSE  0xFE
#define I2C_SLAVE_ADDRESS 0x50 //80 DEC
#define ANLOGPIN 0
bool sendingJson = false;
int sendingNumOfSensors = 0;
byte len = 0;
String json = "";
unsigned long ms = 0;
String labels[]={"l"};
const int numberOfSensors = sizeof(labels)/sizeof(String);
int values[numberOfSensors];

void setup() {
pinMode(4, INPUT);
  pinMode (3, OUTPUT);
  tws_delay(250);
  ms=millis();
  fillValuesArray();
```

```
 TinyWireS.begin(I2C_SLAVE_ADDRESS);     // init I2C Slave mode
 TinyWireS.onRequest(requestEvent);
 fillValuesArray();
 digitalWrite(3, LOW);
}
void loop() {
TinyWireS_stop_check();
}
void requestEvent(){
 fillValuesArray();
 if(sendingNumOfSensors==0){
  TinyWireS.send(numberOfSensors);
  sendingNumOfSensors = 1;
  return;
 }

 if(!sendingJson) {
   createJson(sendingNumOfSensors);
   len = json.length();
   TinyWireS.send(len);
   sendingJson = true;
   return;
  } else {
   digitalWrite(3, HIGH);
   int con = 0;
           char jC[16];
           json.toCharArray(jC, len+1);
   while(con < len){
            TinyWireS.send(jC[con]);
    con++;
   }
   sendingJson = false;
   tws_delay(500);
   digitalWrite(3, LOW);
   sendingNumOfSensors++;
```

```
      if(sendingNumOfSensors > numberOfSensors){
        sendingNumOfSensors = 0;
      }
    }
}
void createJson(int index){
  index-=1;
  json = ",";
  json += labels[index];
  json += ":";
  json += values[index];
}
void fillValuesArray(){
	values[0] = analogRead(ANLOGPIN);
}
```

- Codigo HV Programmer (serial 9600 usar “enter” para escrever)

```
// AVR High-voltage Serial Programmer
// Desired fuse configuration
#define  HFUSE  0x5F   //DF  Defaults for ATtiny25/45/85
#define  LFUSE  0xFE  //62
#define  RST     13    // Output to level shifter for !RESET from transistor to Pin 1
#define  CLKOUT  12    // Connect to Serial Clock Input (SCI) Pin 2
#define  DATAIN  11    // Connect to Serial Data Output (SDO) Pin 7
#define  INSTOUT 10    // Connect to Serial Instruction Input (SII) Pin 6
#define  DATAOUT  9    // Connect to Serial Data Input (SDI) Pin 5
#define  VCC      8    // Connect to VCC Pin 8
int inByte = 0;         // incoming serial byte Computer
int inData = 0;         // incoming serial byte AVR

void setup()
{
  // Set up control lines for HV parallel programming
  pinMode(VCC, OUTPUT);
  pinMode(RST, OUTPUT);
  pinMode(DATAOUT, OUTPUT);
  pinMode(INSTOUT, OUTPUT);
  pinMode(CLKOUT, OUTPUT);
  pinMode(DATAIN, OUTPUT);  // configured as input when in programming mode

  // Initialize output pins as needed
  digitalWrite(RST, HIGH);  // Level shifter is inverting, this shuts off 12V

  // start serial port at 9600 bps:
  Serial.begin(9600);

  establishContact();  // send a byte to establish contact until receiver responds

}

void loop()
{
  // if we get a valid byte, run:
  if (Serial.available() > 0) {
    // get incoming byte:
    inByte = Serial.read();
    Serial.println(byte(inByte));
    Serial.println("Entering programming Mode\n");
    // Initialize pins to enter programming mode
    pinMode(DATAIN, OUTPUT);  //Temporary
    digitalWrite(DATAOUT, LOW);
    digitalWrite(INSTOUT, LOW);
    digitalWrite(DATAIN, LOW);
    digitalWrite(RST, HIGH);  // Level shifter is inverting, this shuts off 12V

    // Enter High-voltage Serial programming mode
    digitalWrite(VCC, HIGH);  // Apply VCC to start programming process
    delayMicroseconds(20);
    digitalWrite(RST, LOW);   //Turn on 12v
    delayMicroseconds(10);
    pinMode(DATAIN, INPUT);   //Release DATAIN
    delayMicroseconds(300);

    //Programming mode

    readFuses();

    //Write hfuse
    Serial.println("Writing hfuse");
    shiftOut2(DATAOUT, INSTOUT, CLKOUT, MSBFIRST, 0x40, 0x4C);
    shiftOut2(DATAOUT, INSTOUT, CLKOUT, MSBFIRST, HFUSE, 0x2C);
    shiftOut2(DATAOUT, INSTOUT, CLKOUT, MSBFIRST, 0x00, 0x74);
    shiftOut2(DATAOUT, INSTOUT, CLKOUT, MSBFIRST, 0x00, 0x7C);

    //Write lfuse
    Serial.println("Writing lfuse\n");
    shiftOut2(DATAOUT, INSTOUT, CLKOUT, MSBFIRST, 0x40, 0x4C);
    shiftOut2(DATAOUT, INSTOUT, CLKOUT, MSBFIRST, LFUSE, 0x2C);
    shiftOut2(DATAOUT, INSTOUT, CLKOUT, MSBFIRST, 0x00, 0x64);
    shiftOut2(DATAOUT, INSTOUT, CLKOUT, MSBFIRST, 0x00, 0x6C);

    readFuses();

    Serial.println("Exiting programming Mode\n");
    digitalWrite(CLKOUT, LOW);
    digitalWrite(VCC, LOW);
    digitalWrite(RST, HIGH);   //Turn off 12v
  }
}


void establishContact() {
  while (Serial.available() <= 0) {
    Serial.println("Enter a character to continue");   // send an initial string
  delay(1000);
 }
}

int shiftOut2(uint8_t dataPin, uint8_t dataPin1, uint8_t clockPin, uint8_t bitOrder, byte val, byte val1)
{
 int i;
    int inBits = 0;
    //Wait until DATAIN goes high
    while (!digitalRead(DATAIN));

    //Start bit
    digitalWrite(DATAOUT, LOW);
    digitalWrite(INSTOUT, LOW);
    digitalWrite(clockPin, HIGH);
  digitalWrite(clockPin, LOW);

 for (i = 0; i < 8; i++)  {

  if (bitOrder == LSBFIRST) {
   digitalWrite(dataPin, !!(val & (1 << i)));
            digitalWrite(dataPin1, !!(val1 & (1 << i)));
        }
  else {
   digitalWrite(dataPin, !!(val & (1 << (7 - i))));
            digitalWrite(dataPin1, !!(val1 & (1 << (7 - i))));
        }
        inBits <<=1;
        inBits |= digitalRead(DATAIN);
        digitalWrite(clockPin, HIGH);
  digitalWrite(clockPin, LOW);

 }
```

```
    //End bits
    digitalWrite(DATAOUT, LOW);
    digitalWrite(INSTOUT, LOW);
    digitalWrite(clockPin, HIGH);
     digitalWrite(clockPin, LOW);
    digitalWrite(clockPin, HIGH);
    digitalWrite(clockPin, LOW);

    return inBits;
}

void readFuses(){
   //Read lfuse
  shiftOut2(DATAOUT, INSTOUT, CLKOUT, MSBFIRST, 0x04, 0x4C);
  shiftOut2(DATAOUT, INSTOUT, CLKOUT, MSBFIRST, 0x00, 0x68);
  inData = shiftOut2(DATAOUT, INSTOUT, CLKOUT, MSBFIRST, 0x00, 0x6C);
  Serial.print("lfuse reads as ");
  Serial.println(inData, HEX);

  //Read hfuse
  shiftOut2(DATAOUT, INSTOUT, CLKOUT, MSBFIRST, 0x04, 0x4C);
  shiftOut2(DATAOUT, INSTOUT, CLKOUT, MSBFIRST, 0x00, 0x7A);
  inData = shiftOut2(DATAOUT, INSTOUT, CLKOUT, MSBFIRST, 0x00, 0x7E);
  Serial.print("hfuse reads as ");
  Serial.println(inData, HEX);

  //Read efuse
  shiftOut2(DATAOUT, INSTOUT, CLKOUT, MSBFIRST, 0x04, 0x4C);
  shiftOut2(DATAOUT, INSTOUT, CLKOUT, MSBFIRST, 0x00, 0x6A);
  inData = shiftOut2(DATAOUT, INSTOUT, CLKOUT, MSBFIRST, 0x00, 0x6E);
  Serial.print("efuse reads as ");
  Serial.println(inData, HEX);
  Serial.println();
}
```

- Ligações

ESP-8266

D1-L1(level shifter)

D2-L2

ATtynni85

P1-Resistencia10k (GND) ,LDR(VCC)

P2,P3-Cristal 16 MHz

P4-GND (ESP)

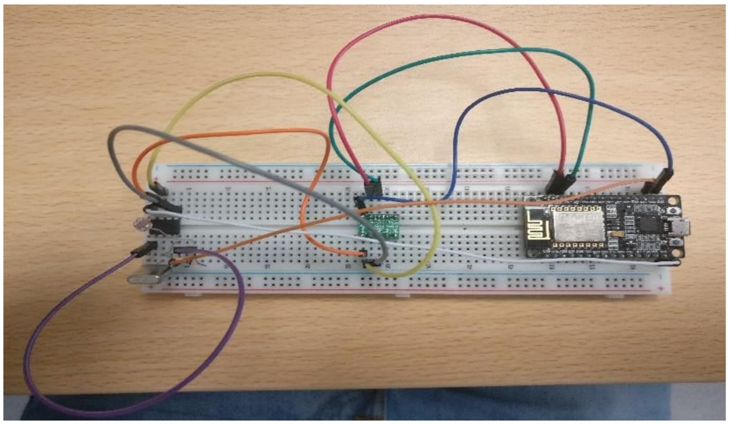
P5-Vcc (ESP-VIN)

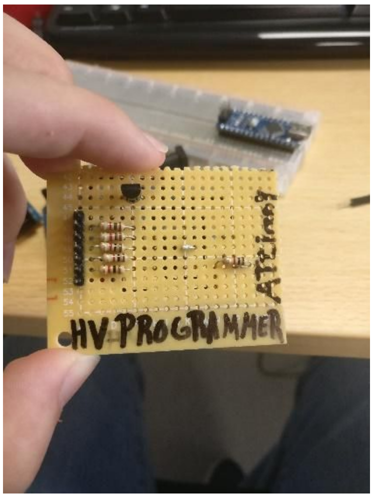
P6-H1 (level shifter)

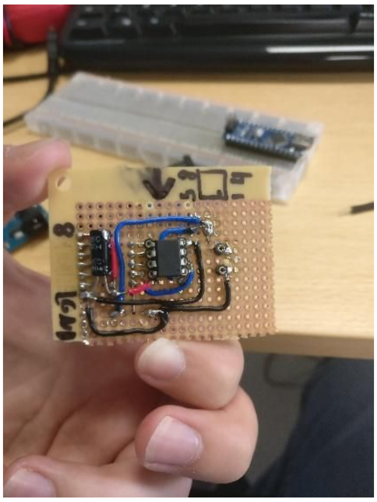
P7-Não usado pode servir como pino digital

P8-H2

- Fotos







- Fontes

https://www.youtube.com/watch?v=yAT_TdD6nL0

https://arduinodiy.wordpress.com/2015/05/16/high-voltage-programmingunbricking-for-attiny/

http://www.atmel.com/images/atmel-2586-avr-8-bit-microcontroller-attiny25-attiny45-attiny85_datasheet.pdf

http://www.engbedded.com/fusecalc/

http://www.martyncurrey.com/arduino-atmega-328p-fuse-settings/

https://github.com/rambo/TinyWire

https://github.com/damellis/attiny

http://www.technoblogy.com/show?LSE

http://www.instructables.com/id/How-to-unlock-Digispark-ATtiny85-and-convert-it-to/