



Relatório de Actividades

Desenho e Construção de uma Estação Hidrográfica Telemétrica

2019

Kishan Nareshpal Jadav
Eng. Gorka Solana Arteché

Introdução

A necessidade de monitoramento dos rios surge como uma das formas de prevenção e de alerta antecipada contra as inundações e cheias que poderão ocorrer em zonas baixas, afectando várias populações, assim como às infra estruturas ao redor. Utiliza-se até então, em alguns locais, uma régua milimetrada que para ser manuseada, obrigando assim a locomoção para estes lugares de forma a fazer as devidas medições.

O desenvolvimento deste sistema de monitoramento autónomo com a transmissão de informações em tempo real não obrigam a locomoção para alguns destes locais remotos ou de difícil acesso e também obtém-se uma maior quantidade de dados sobre o local.

O desenho e construção desta estação hidrográfica telemétrica, consiste em 9 fases de actividades:

- Revisão literária
- Aquisição
- Montagem do protótipo
- Programação
- Calibração
- Configuração do Servidor
- Provas Preliminares
- Implementação
- Monitoramento

1. Aquisição

Qntd.	Descrição	Valor pago
1	Adafruit / Clock & Timer	\$13.99
1	Adafruit RF FONA 808 Shield	\$15.10
1	Adafruit Energy Harvesting modules Huge 6V Solar Panel	\$61.55

1	AAEON / Battery Packs	\$28.08
1	Adafruit Multiple Function Weatherproof Ultrasonic Rangefinder	\$110.77
1	Adafruit Sensor BME680	\$22.57
1	Arduino UNO R3 AVR	\$22.07
5	Cintas Plásticas	50 MT
1	Multímetro	1950 MT
1	União red 31x1/2	35 MT
1 rolo	Fita isoladora	75 MT
1	Bateria 6V	600 MT
1	Caixa Estanque IP66	675 MT
6m	Fio electrico condutor black	475 MT
1	Tigela cor azul Gnd/	320 MT
1	Estanho de 0,5 mm	415 MT
1	Ferro de Solda 127V	3,360 MT
1	Buscapolo Red line	125 MT
1	Recarga da Vodacom	2,000 MT
1 rolo	Mounting Tape/Fita cola de duas caras	375 MT
	TOTAL PAGO	34,407.2 MT

Fez-se a compra online, através da Mouser Electronics no dia 13 de Outubro de 2018, de alguns dos componentes necessários na montagem do nosso protótipo avaliados em um total de 23,452.20 MT (\$339.13 USD) e de outros componentes que foram adquiridos em lojas locais de venda a retalho, com o valor restante. Ficando assim com um **total gasto** de 34,407.20 MT e 592.8 MT como a sobra.

2. Provas e Montagem do protótipo

Em princípio, fizeram-se testes de todos os componentes electrónicos comprados que serão montados.

Começou-se com o teste do Arduino UNO, que é um **microcontrolador** composto por circuitos de entrada/saída e que pode ser facilmente conectado à um computador e programado através do **Arduino IDE** (*Integrated Development Environment*, ou *Ambiente de Desenvolvimento Integrado*) utilizando uma linguagem de programação baseada em C/C++, sem a necessidade de equipamentos extras além de um cabo USB. Fez-se a conexão deste a um computador, através de um cabo USB e o software de programação do Arduino, **Arduino IDE**, que a partir de agora irei me referir somente de **IDE**.

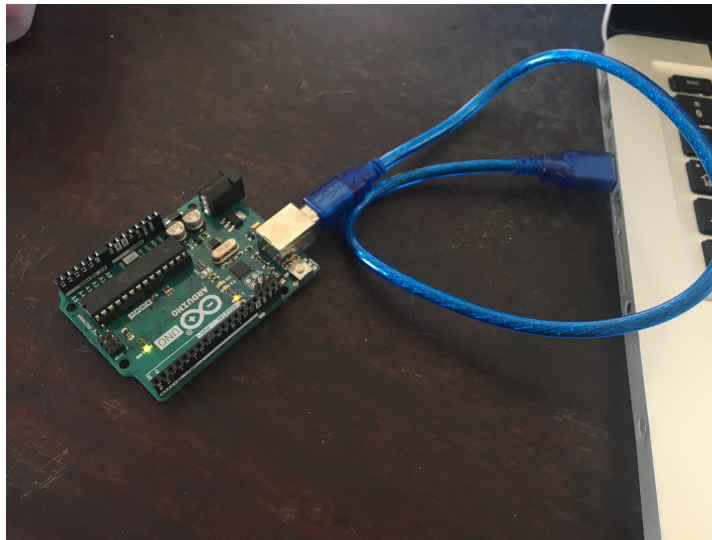
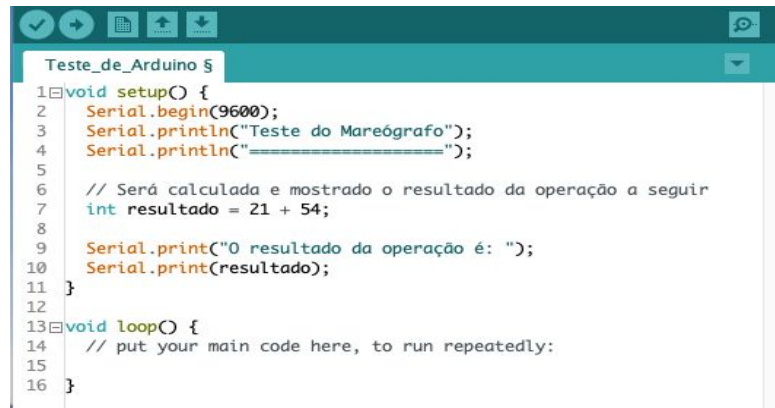


Figura 1: Conexão do Arduino ao computador

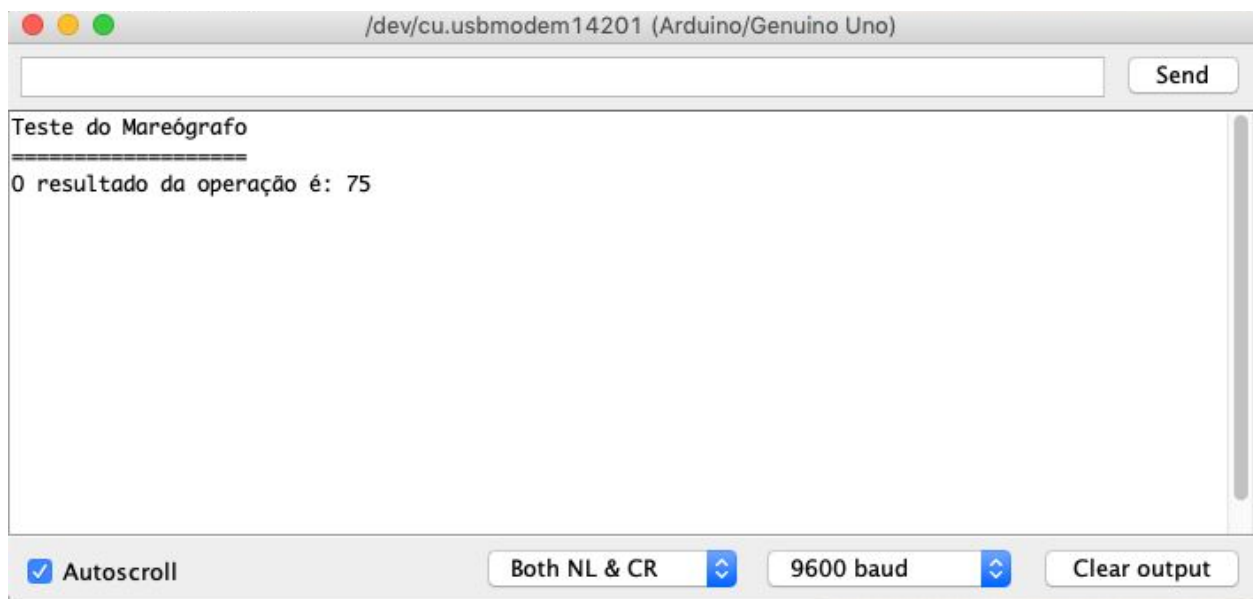
Para verificar se este microcontrolador está em condições de ser programado, escreveu-se e transferiu-se o seguinte código para o teste:



```
1 void setup() {
2   Serial.begin(9600);
3   Serial.println("Teste do Mareógrafo");
4   Serial.println("=====");
5
6   // Será calculada e mostrado o resultado da operação a seguir
7   int resultado = 21 + 54;
8
9   Serial.print("O resultado da operação é: ");
10  Serial.print(resultado);
11 }
12
13 void loop() {
14   // put your main code here, to run repeatedly:
15 }
16 }
```

Figura 2: Código para teste escrito no ambiente de trabalho do Arduino (Arduino IDE)

O código acima, tem a função de imprimir o resultado de uma operação matemática simples. O que é suficiente para verificar se o arduino está em boas condições. Após a escrita do código, transferiu-se o código para o Arduino, no que resultou em uma resposta correta e esperada:



```
/dev/cu.usbmodem14201 (Arduino/Genuino Uno)

Teste do Mareógrafo
=====
O resultado da operação é: 75

Autoscroll Both NL & CR 9600 baud Clear output
```

Figura 3: O resultado da operação está correcta, concluindo assim de que o Arduino está funcional.

A seguir, utilizando uma placa de ensaio (*protoboard*), testou-se o sensor Adafruit BME 680 de gás, temperatura, humidade e pressão.

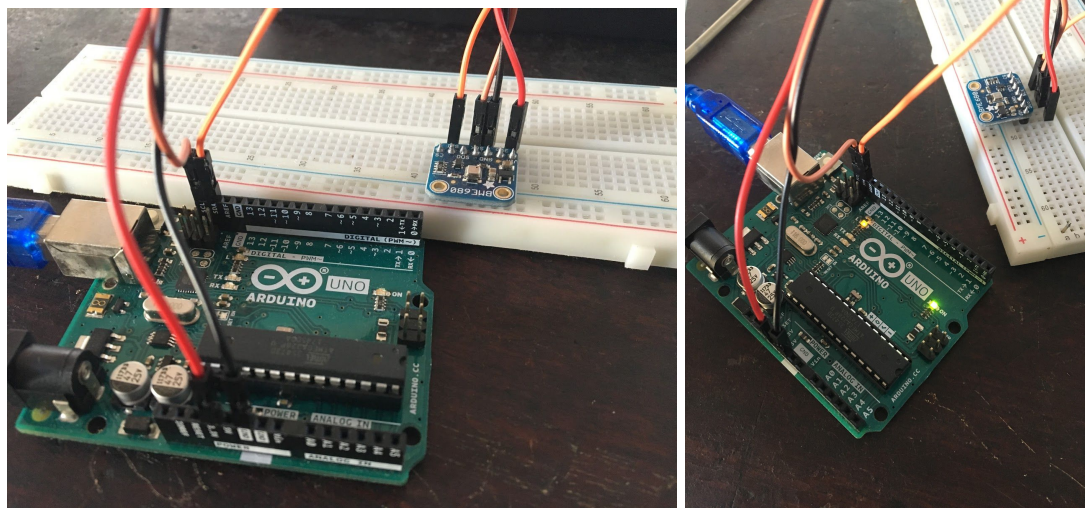


Figura 4: Arduino UNO e o Sensor BME 680 conectados através do protoboard.

Transferimos o seguinte código para o arduino de modo a testar o componente Adafruit BME. O código seguinte tem como objectivo imprimir os seguintes dados coletados dos sensores que estão acoplados no componente a cada 2 segundos.

- Temperatura
- Quantidade de gás
- Pressão
- Humidade
- Altitude aproximada

```
/*  
  BME 680 Teste  
*/  
  
#include <Wire.h>  
#include <SPI.h>  
#include <Adafruit_Sensor.h>  
#include "Adafruit_BME680.h"  
  
#define BME_SCK 13  
#define BME_MISO 12  
#define BME_MOSI 11  
#define BME_CS 10
```

```

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME680 bme; // I2C
//Adafruit_BME680 bme(BME_CS); // hardware SPI
//Adafruit_BME680 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK);

void setup() {
  Serial.begin(9600);
  while (!Serial);
  Serial.println(F("BME680 test"));

  if (!bme.begin()) {
    Serial.println("Could not find a valid BME680 sensor, check wiring!");
    while (1);
  }

  // Set up oversampling and filter initialization
  bme.setTemperatureOversampling(BME680_OS_8X);
  bme.setHumidityOversampling(BME680_OS_2X);
  bme.setPressureOversampling(BME680_OS_4X);
  bme.setIIRFilterSize(BME680_FILTER_SIZE_3);
  bme.setGasHeater(320, 150); // 320*C for 150 ms
}


void loop() {
  if (! bme.performReading()) {
    Serial.println("Failed to perform reading :(");
    return;
  }
  Serial.print("Temperature = ");
  Serial.print(bme.temperature);
  Serial.println(" *C");

  Serial.print("Pressure = ");
  Serial.print(bme.pressure / 100.0);
  Serial.println(" hPa");
}

```

```
Serial.print("Humidity = ");
Serial.print(bme.humidity);
Serial.println(" %");
Serial.print("Gas = ");
Serial.print(bme.gas_resistance / 1000.0);
Serial.println(" KOhms");

Serial.print("Approx. Altitude = ");
Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
Serial.println(" m");
Serial.println();
delay(2000);
}
```



Approx. Altitude = 4.00 m

Temperature = 31.96 *C
Pressure = 1012.77 hPa
Humidity = 64.82 %
Gas = 134.58 KOhms
Approx. Altitude = 4.00 m

Temperature = 31.96 *C
Pressure = 1012.79 hPa
Humidity = 64.76 %
Gas = 137.23 KOhms
Approx. Altitude = 3.83 m

☒ Autoscroll Both NL & CR 9600 baud Clear output

Figura 5: Impressão dos dados do sensor BME 680

Pela qual foi concluído que o Adafruit BME680 está em perfeitas condições de funcionamento e uso. Notando que estes testes foram feitos em um ambiente interno, ou seja, dentro de um quarto e os resultados destes testes poderão variar dependendo das condições que se encontrará no campo onde se aplicará o protótipo.

A seguir, fez-se o teste do Sensor Ultrassónico Maxbotix 7092 que, através de cabos, conectado ao Arduino directamente sem o uso de Protoboard. E o arduino por sua vez, conectado ao computador. Concluindo de que o sensor esta funcional.

O código teste transferido ao arduino tem a função de imprimir a distância do objecto mais próximo a este sensor.

```
/* Maxbotix simple test */
#include "Maxbotix.h"

Maxbotix rangeSensorAD(A0, Maxbotix::AN, Maxbotix::LV);

void setup() {
  Serial.begin(9600);
}

void loop() {
  unsigned long start;
  Serial.println("Reading...");
  // AD
  start = millis();
  Serial.print("AD: ");
  Serial.print(rangeSensorAD.getRange());
  Serial.print("cm - ");
  Serial.print(millis() - start);
  Serial.println("ms");
  Serial.println();
  delay(5000);
}
```

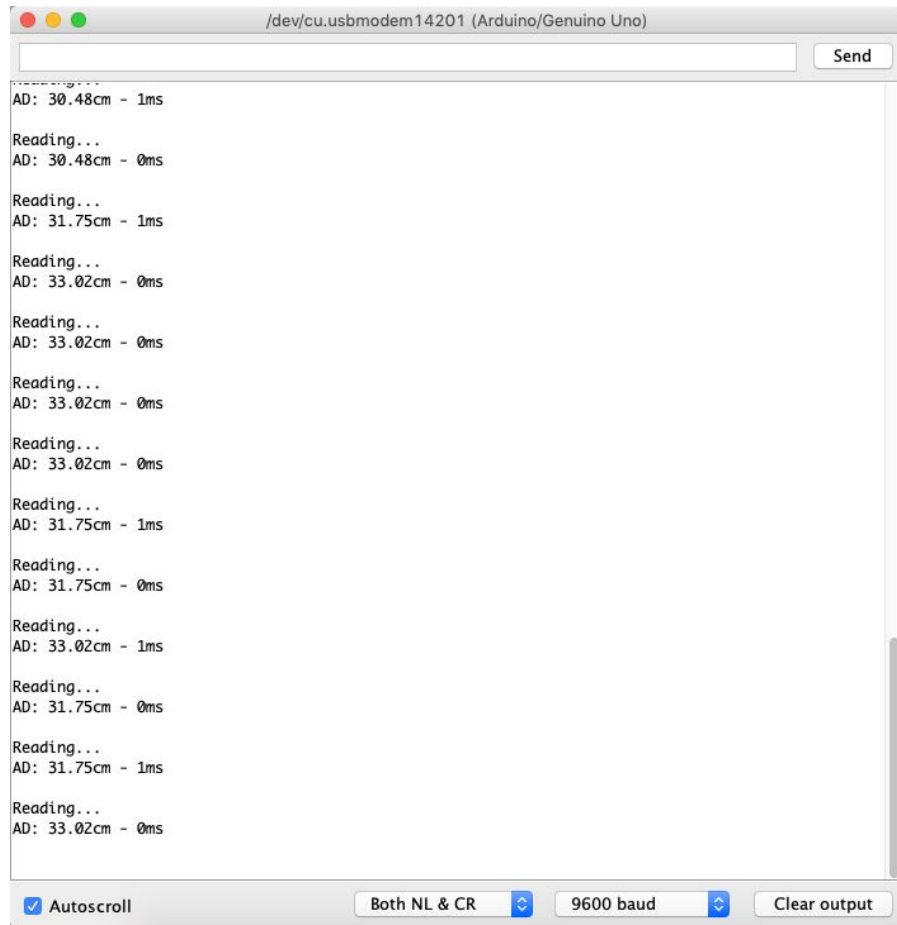
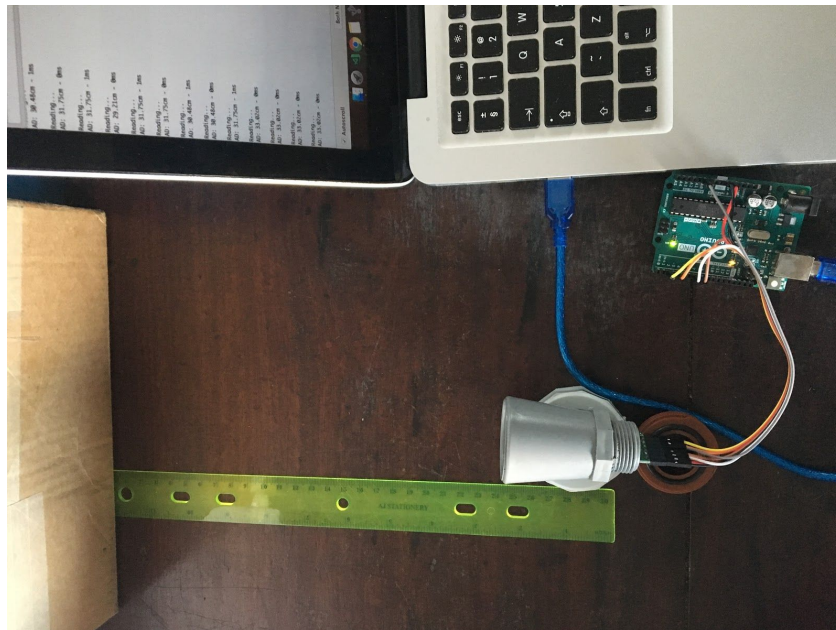
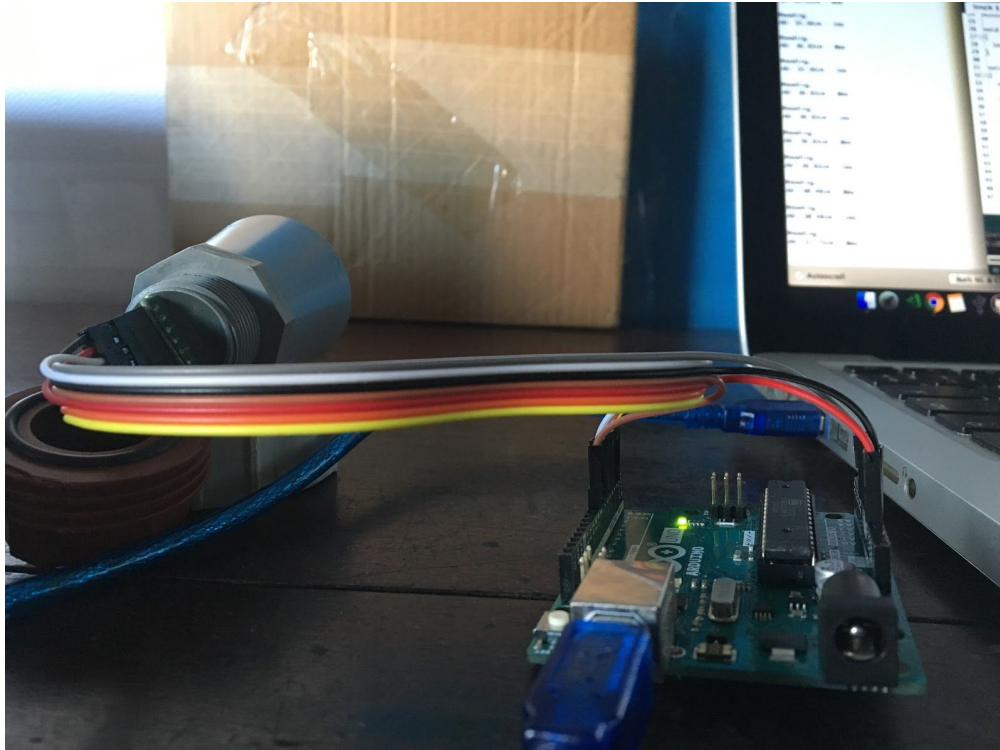


Figura 5: Impressão da distância coletada pelo sensor a cada 5 segundos em Centímetros.



Figuras 6 e 7: Conexão do sensor ultrasónico ao arduino e ao computador em fases de teste. E uma régua de 30 cm para o relacionamento.

Em seguida testou-se o módulo FONA GSM GPS SHIELD SIM808. Inseriu-se um cartão SIM da operadora Vodacom Moçambique, com uma bateria de 6V 4.5Ah. Este módulo foi acoplado a placa do Arduino UNO, que por sua vez está conectado ao computador. Sem a utilização do protoboard. Para testes preliminares, transferiu-se o seguinte código que tem por objectivo imprimir a localização do sensor GPS incluído nesse *Shield*.

```
/**
 * Teste do Adafruit FONA Shield 808
 */
#include "Adafruit_FONA.h"

// standard pins for the shield, adjust as necessary
#define FONA_RX 2
#define FONA_TX 3
#define FONA_RST 4
#include <SoftwareSerial.h>

SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
SoftwareSerial *fonaSerial = &fonaSS;

Adafruit_FONA fona = Adafruit_FONA(FONA_RST);

void setup() {

    while (! Serial);

    Serial.begin(115200);

    Serial.println(F("Initializing FONA... (May take a few seconds)"));

    fonaSerial->begin(4800);
    if (! fona.begin(*fonaSerial)) {
        Serial.println(F("Couldn't find FONA"));
        while(1);
    }
}
```

```

}
Serial.println(F("FONA is OK"));
// Try to enable GPRS

Serial.println(F("Enabling GPS..."));
fona.enableGPS(true);
}

void loop() {
    delay(2000);

    float latitude, longitude, speed_kph, heading, speed_mph, altitude;

    // if you ask for an altitude reading, getGPS will return false if
    // there isn't a 3D fix
    boolean gps_success = fona.getGPS(&latitude, &longitude,
    &speed_kph, &heading, &altitude);

    if (gps_success) {

        Serial.print("GPS lat:");
        Serial.println(latitude, 6);
        Serial.print("GPS long:");
        Serial.println(longitude, 6);
        Serial.print("GPS speed KPH:");
        Serial.println(speed_kph);
        Serial.print("GPS speed MPH:");
        speed_mph = speed_kph * 0.621371192;
        Serial.println(speed_mph);
        Serial.print("GPS heading:");
        Serial.println(heading);
        Serial.print("GPS altitude:");
        Serial.println(altitude);

    } else {
        Serial.println("Waiting for FONA GPS 3D fix...");
    }
}

```

```

// Fona 3G doesnt have GPRSlocation :/
if ((fona.type() == FONA3G_A) || (fona.type() == FONA3G_E))
    return;
// Check for network, then GPRS
Serial.println(F("Checking for Cell network..."));
if (fona.getNetworkStatus() == 1) {
    // network & GPRS? Great! Print out the GSM location to compare
    boolean gsmloc_success = fona.getGSMLoc(&latitude, &longitude);

    if (gsmloc_success) {
        Serial.print("GSMLoc lat:");
        Serial.println(latitude, 6);
        Serial.print("GSMLoc long:");
        Serial.println(longitude, 6);
    } else {
        Serial.println("GSM location failed...");
        Serial.println(F("Disabling GPRS"));
        fona.enableGPRS(false);
        Serial.println(F("Enabling GPRS"));
        if (!fona.enableGPRS(true)) {
            Serial.println(F("Failed to turn GPRS on"));
        }
    }
}
}
}
}

```

```

/dev/cu.usbmodem14201 (Arduino/Genuino Uno)
Send

Adafruit FONA 808 & 3G GPS demo
Initializing FONA... (May take a few seconds)
Attempting to open comm with ATs
---> AT
<---
---> AT
<---
---> AT
<---
---> AT
<---
---> AT
<---
---> AT
<---
---> ATE0
<--- ATE0
---> ATE0
<--- OK
---> AT+CVHU=0
<--- OK
---> ATI
<--- SIM808 R14.18

OK

---> AT+CPMS="SM","SM","SM"
<--- ERROR
FONA is OK
Enabling GPS...
---> AT+CGNSPWR?
<--- +CGNSPWR: 0
---> AT+CGNSPWR=1
<--- OK
---> AT+CGNSINF
<--- +CGNSINF: 1,0,19800105235944.000,,0.00,0.0,0,,,,,0,0,,,,
Waiting for FONA GPS 3D fix...
Checking for Cell network...
---> AT+CREG?

```

Figura 8: Ao que, concluimos que o componente está funcional e pronto para ser usado.

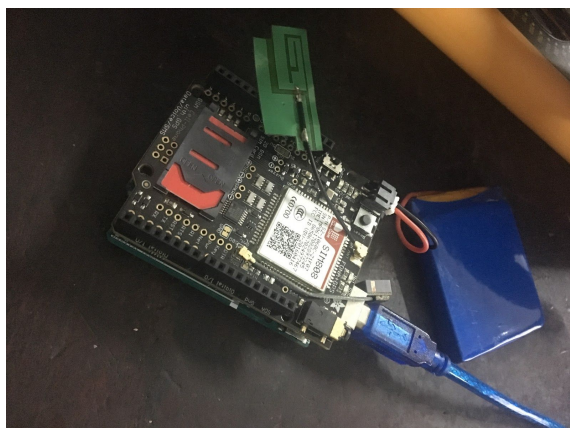


Figura 9: Placa FONA GSM SIM808 acoplado ao Arduino Uno e uma bateria de 6V. Conectados com um cabo USB ao computador

A seguir, foi feito o teste do painel solar, que foi ligado em paralelo a uma Bateria do tipo 6V 4.5Ah e ao Arduino UNO. Que serve como fonte de energia renovável ao microcontrolador.

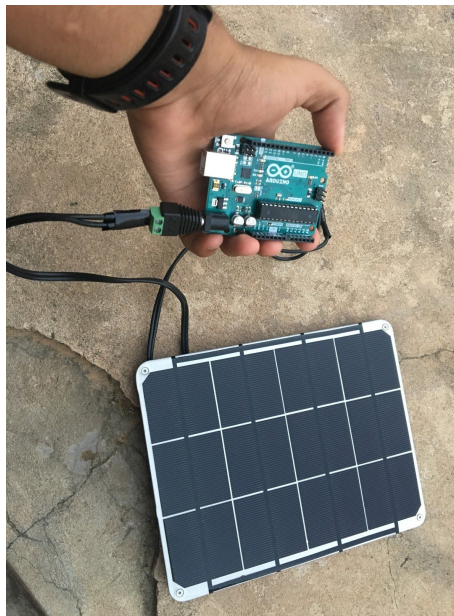
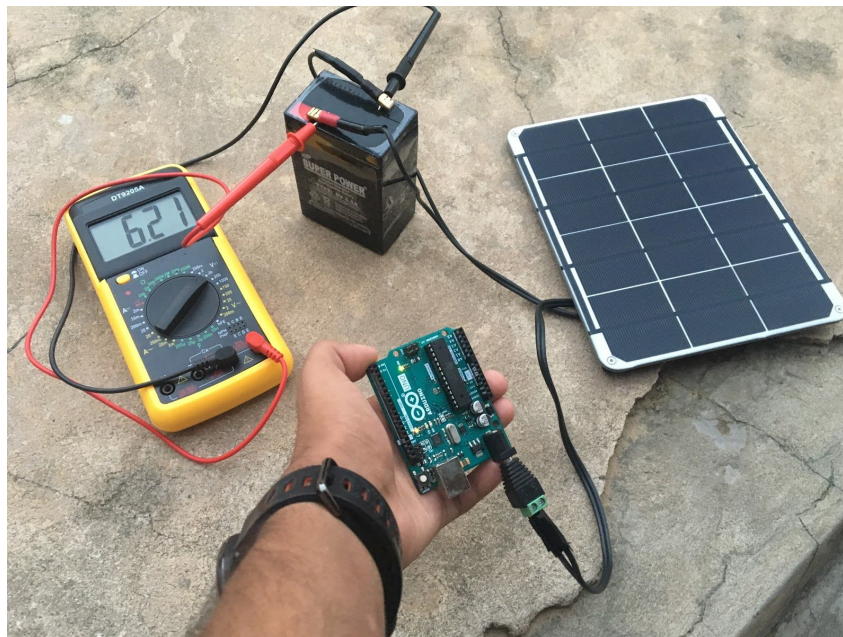


Fig. 10



Figuras 10 e 11: Arduino UNO conectado directamente ao painel solar à luz de dia (Fig10). Arduino UNO, bateria de 6v 4.5Ah e painel solar conectados em paralelo. Multímetro para medir a corrente. (Fig. 11)

E por fim, para testar a estanqueidade da caixa que irá armazenar os componentes contra o sol, vento e águas dos rios. Submergiu-se a caixa à uma profundidade de 10cm em um balde cheio de água durante 1 minutos para verificar a marcação internacional de proteção IP66. Na qual foi verificada que de facto a caixa é estanque e não infiltra água por dentro. Pois uma partícula pequena de água poderá danificar os componentes electrónicos que estarão dentro da caixa.

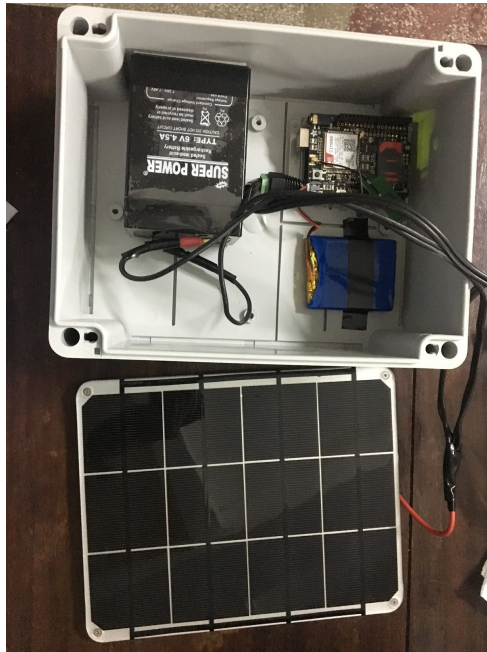


Figura 12 e 13: Estrutura interna da caixa estanque IP66. Caixa estanque submersa a uma profundidade de 30 cm.

Após os testes realizados dos componentes, iniciou-se a montagem do protótipo. O protótipo foi desenhado para ser à prova de água. Uma caixa estanque IP66 foi usada para alocar todo o equipamento electrónico internamente, fora do alcance das águas dos rios.

Na parte interna da caixa foram afixados o microcontrolador Arduino UNO, a bateria de 6V recarregável que irá armazenar a carga durante o período da manhã através do painel solar, para ser utilizada durante as noites. Estes foram afixados com uma fita cola de duas caras, para evitar os movimento perturbadores nas águas dos rios onde serão colocados.

O sensor ultrasónico (à prova de água), responsável pela medição do nível da água, assim como o painel solar foram colocados por fora da caixa para uma melhor performance dos dois. Conectados isoladamente para dentro da caixa, ao microcontrolador Arduino responsável pela computação, e à bateria de 6V.\



Inhambane, 15 de Fevereiro de 2019

Kishan Nareshpal Jadav