```
#include "Adafruit_FONA.h"


#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);

float Measured = 0;

int measurePin = A0;

void status_gsm()
{
  Serial.println("AT");
  fonaSS.println("AT");
  if(fonaSS.find("OK"))
  {
    Serial.println("Connected");
  }
  else
  {
    Serial.println("Connection failed");
  }
  delay(100);
}

 void setting_SIM()
{
  Serial.println(("AT+CREG=1")); //  Registration status    Response -  0 - not registered, 1 - registered, 2 -  not registered
  fonaSS.println(("AT+CREG=1"));
  delay(500);
  Serial.println(fonaSS.readString());

  Serial.println("AT+CIPMODE=0"); // Selecting TCPIP application mode. '0' os non-transparent mode and '1' is transparent mode.
  fonaSS.println("AT+CIPMODE=0");
  delay(300);
  Serial.println(fonaSS.readString());

  Serial.println(("AT+CIPSHUT"));// Close the GPRS PDP context
  fonaSS.println(("AT+CIPSHUT"));
  delay(1000);
  Serial.println(fonaSS.readString());

  Serial.println(("AT+CIPMUX=0")); // AT+CIPMUX AT command configures the device for a single or multi IP
connection,AT+CIPMUX= - Value can be '0' (single IP) or '1' (multi IP)
  fonaSS.println(("AT+CIPMUX=0"));
  delay(500);
  Serial.println(fonaSS.readString());

  Serial.println(("AT+CGATT=1")); // attach or detach the device to packet domain service,Responses  -   +CGATT=[]  OK ERROR
  fonaSS.println(("AT+CGATT=1"));
  delay(500);
```

```
  Serial.println(fonaSS.readString());

  Serial.println(("AT+CSTT=\"WHOLESALE\"")); //  AT command sets up the apn, user name and password for the PDP context.
  fonaSS.println(("AT+CSTT=\"WHOLESALE\"")); // +CSTT: "APN","USER","PWD"
  delay(3000);
  fonaSS.println(("AT+CSTT?"));
  delay(3000);
  Serial.println(fonaSS.readString());

  Serial.println("AT+SAPBR=3,1,\"APN\",\"wholesale\"");
  fonaSS.println("AT+SAPBR=3,1,\"APN\",\"wholesale\"");//setting the APN, the second need you fill in your local apn server
  delay(1000);
  Serial.println(fonaSS.readString());


  Serial.println(("AT+CIICR")); // Brings up the GPRS
  fonaSS.println(("AT+CIICR"));
  delay(3000);
  Serial.println(fonaSS.readString());

  Serial.println(("AT+CIFSR"));
  fonaSS.println(("AT+CIFSR")); //request IP
  delay(1000);
  Serial.println(fonaSS.readString());

}

void setup()
{
  fonaSS.begin(4800);
  Serial.begin(115200);
  delay(2000);
  Serial.print("Status:");
  status_gsm();  // Check connection
  setting_SIM(); // Setup procedure

  while (! Serial);
  Serial.println("Initializing FONA");

  fonaSerial->begin(4800);
  if (! fona.begin(*fonaSerial)) {
    Serial.println("Finding FONA");
    while(1);
  }

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

}

void connection()
{

 delay(2000);

  float latitude, longitude, altitude;

  boolean gps_success = fona.getGPS(&latitude, &longitude, &altitude);

  if (gps_success) {

    Serial.print("GPS lat:");
```

```
  Serial.println(latitude, 6);
  Serial.print("GPS long:");
  Serial.println(longitude, 6);
  Serial.print("GPS altitude:");
  Serial.println(altitude);

 } else {
  Serial.println("Applying 3D Fix");
 }

 Serial.println(("AT+CIPSTART=\"TCP\",\"184.106.153.149\",\"80\""));
 fonaSS.println(("AT+CIPSTART=\"TCP\",\"184.106.153.149\",\"80\"")); // Comamnds starts a TCP or UDP connection
 delay(6000);
 Serial.println(fonaSS.readString());
 delay(1000);

 Serial.println(("AT+CIPSEND")); //  AT command is used to send the data over the TCP or UDP connection
 fonaSS.println(("AT+CIPSEND"));
 delay(2000);


 Serial.print("GET /update?key=8BF5XVD72MFCVWS0&field1=" + String(Measured) + "&field2="+String(latitude,6)+
"&field3="+String(longitude,6));    // Request to send the data
 fonaSS.print("GET /update?key=8BF5XVD72MFCVWS0&field1=" + String(Measured) + "&field2="+String(latitude,6)+
"&field3="+String(longitude,6));
 delay(8000);
 Serial.println(fonaSS.readString());
 delay(1000);

 Serial.print(" HTTP/1.1\r\n");   // Fetches the new page
 fonaSS.print(" HTTP/1.1\r\n");  // Responses are important
 Serial.println(fonaSS.readString());
 delay(1000);

 Serial.print("Host: 184.106.153.149\r\n");
 fonaSS.print("Host: 184.106.153.149\r\n");
 Serial.println(fonaSS.readString());
 delay(1000);


 fonaSS.print("Connection: keep-alive");  // Connection general header controls whether or not the network connection stays open
after the current transaction finishes, value sent is keep-alive, the connection is persistent and not closed
 fonaSS.print("Connection: close");      //   Indicates that either the client or the server would like to close the connection.
 fonaSS.print("\r\n");
 fonaSS.print("\r\n");
 fonaSS.println();
 Serial.println(fonaSS.readString());
 delay(1000);

 fonaSS.println((char)26);
 delay(1000);

  if(fonaSS.find("SEND OK"))
  {
    delay(1000);
    while (fonaSS.available())
    {
      Serial.println(fonaSS.readString()); //feedback data
    }
    fonaSS.println(("AT+CIPCLOSE"));
  }
  else
```

```
   {
     connection();
   }
  }



void loop() {

long Int1=0;
long lastVal1 = 0;

    unsigned long currentTime = millis();

  if (currentTime - lastVal1 < Int1) {
    lastVal1 = currentTime;
    Serial.print("Timer of    ");
    Serial.print(lastVal1 / 1000);
    Serial.println(" Sec");
    //Timer
  }


  Measured = analogRead(measurePin);



 Serial.print(Measured);
 Serial.print(",");


    delay(5000);
    connection();
    delay(1000);

  }
```