Solaire

Team EE - GSM Controlled LED

CSM117 Project Report

Team Members:

Rex Taymany — 404841213

Shayan Darian — 904679011

Nick Marks — 904844704

Introduction:

After conducting the Special Wireless Experiments, we thought it would be useful and practical to incorporate wireless technology to relay input information to our circuits rather than having to manually push a button on the circuit itself. By creating a mobile app and incorporating a GSM module, we hoped to learn more about server and network architecture and the processes that encompass each step.

Functionality:

Our app (Solaire) is an iOS based mobile app that allows users to wirelessly control an circuit elements, in this case an LED, through the use of a GSM module connected to an Arduino micro-controller. The use of a GSM over WiFi or Bluetooth allows for more flexibility in distance and location when controlling the circuit.

How to run the project:

The Xcode project and its code should be exported either as a simulation of a device in Xcode or it can be exported to an iOS based device. The code related to the Arduino and GSM device should be installed to the Arduino UNO through the Arduino IDE after all the necessary connections are established. When all components are properly loaded, the app should enable the user to control the LED.

Wireless Technologies:

- GSM
- Ting (ISP)
- dweet.io

Software and Hardware:

- Xcode, Swift, Arduino IDE, Adafruit FONA 800, Antennae, Ting 2G SIM card, Arduino UNO, LED, Lithium Batter, Resistor, Breadboard

GSM:

Global System for Mobile Communication, or GSM for short, is a digital mobile telephony system that is widely used in Europe and other parts of the world. GSM uses a variation of time division multiple access (TDMA) and is the most widely used of the three digital wireless telephony technologies, operating at either the 900 MHz or 1800 MHz frequency band. We chose to use GSM as our choice wireless technology as opposed to WiFi or Bluetooth since we wanted to be able to operate and monitor our circuit from anywhere, whether it be from home or a moving car, over long distances.

Ting (ISP)

In order for our GSM module to work reliably as intended, we used Ting as our ISP to provide the coverage we needed to access our circuit from any remote location. The provided SIM card in installed onboard the GSM module and operates any way a typical phone would operate with standard message rates and fees.

dweet.io

We incorporated the web based RESTful API that dweet.io employs in order to communicate with and monitor our circuit. This allows for more flexibility as the JSON requests can be easily coded on the front end through built in function calls and libraries provided in the Xcode and Swift environment while Adafruit GSM libraries included in the Arduino IDE would be able to handle the reception of these requests.

Implementation:

The app's front end is coded in Swift and utilizes Xcode's mobile development platform to create an interface for user input in the form of a switch and button. The app's code would relay the user input to a web based RESTful API (dweet.io) in the form of JSON requests. These requests would be picked up and communicated by the GSM module to the serial monitor in the Arduino. By reading the incoming stream provided by the GSM module, the Arduino would be able to match strings to the correct functionality intended for the circuit from the user inputs.

Member Contribution:

Rex Taymany — Worked on the the front end UI and implemented the middleware necessary to connect and send information to the web based RESTful API. Configured the backend connection for the GSM to the cellular network.

Shayan Darian — Worked on the GSM to Arduino communication. Configured the necessary functions to send and receive HTTP requests from the module and record the information in the serial monitor.

Nick Marks — Worked on the Arduino to circuits connection. Configured the GSM to parse the incoming JSON response from the GSM module and set up a detection method to carry out the circuits intended functionality.

Conclusions and Future Improvements:

We were able to successfully implement and demonstrate our intended functionality of switching on and off an LED through our design. There is a lag time of about 2 seconds in between user input and LED output; the tradeoff of this lag time is the benefit of being able to operate the app from any location within an ISP's coverage range for greater operating distance and flexibility of location. This project served as a great proof of concept and exposure to wireless technology and mobile app development where future improvements can be easily implemented. One such improvement would be for the circuit to accommodate more circuit elements that would be able to relay information such as LED brightness, sensor information, and connected Arduino port to the app. The app itself could be improved upon to automatically update the UI with controls in the event that elements are added/removed from the circuit.

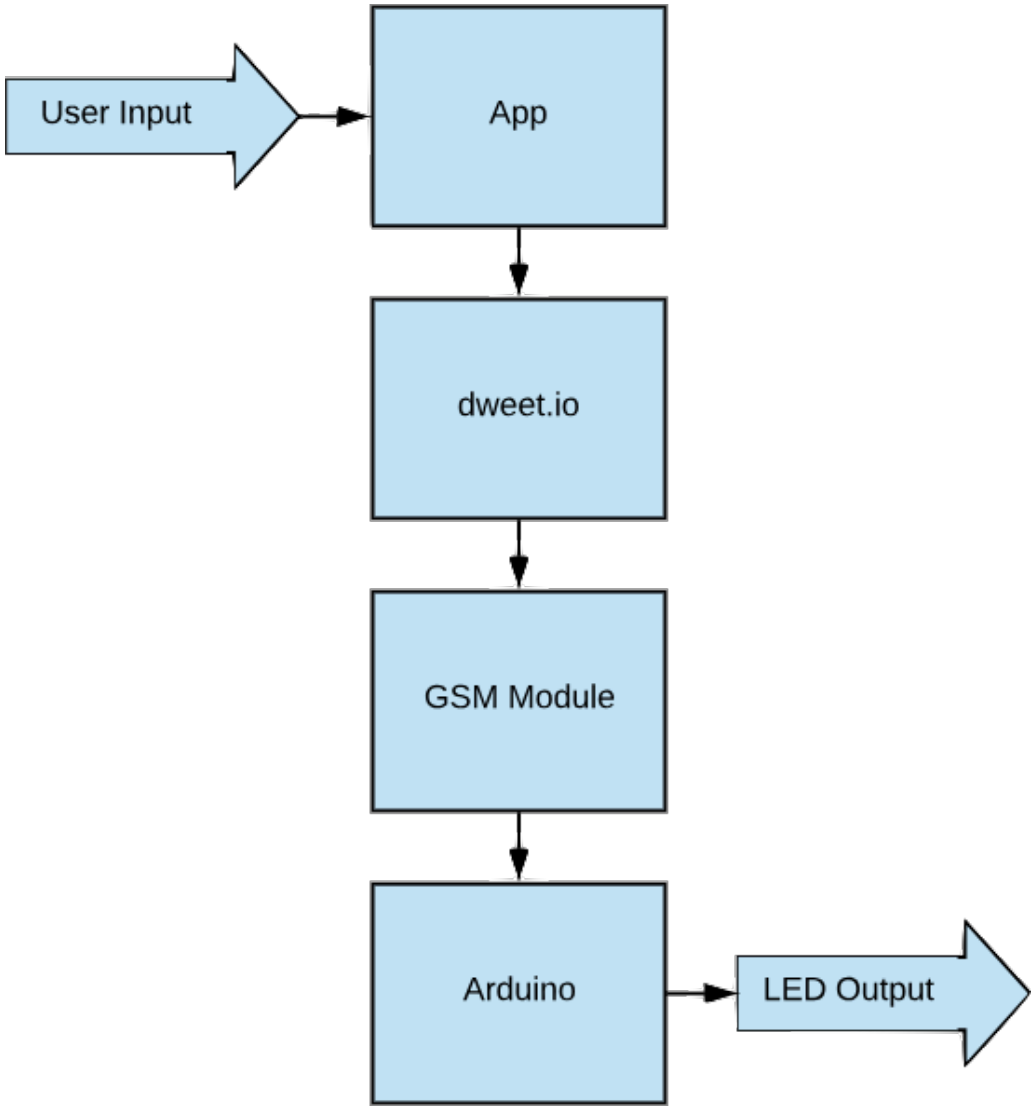
SOLAIRE

TEAM EE

REX TAYMANY, SHAYAN DARIAN, NICK MARKS

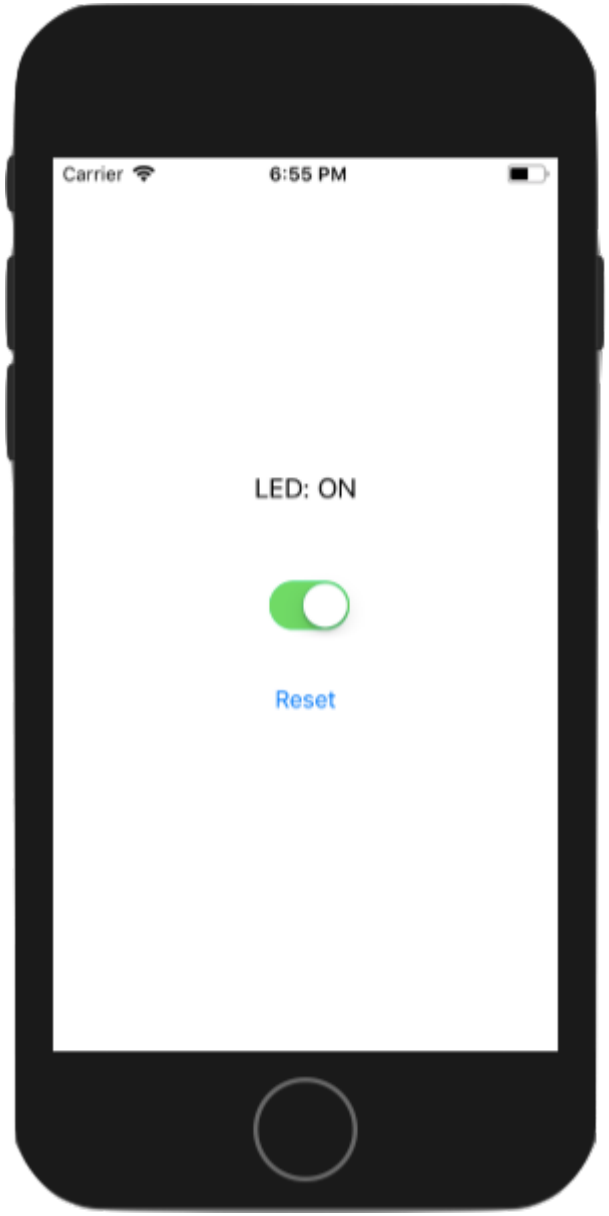
OVERVIEW

- Solaire is an iOS based app that communicates with a GSM cellular module in conjunction with an Arduino that allows users to control an onboard LED
- Our goal for this project was to implement wireless technology to our circuits in place of manual switches and buttons with a simple app to control it



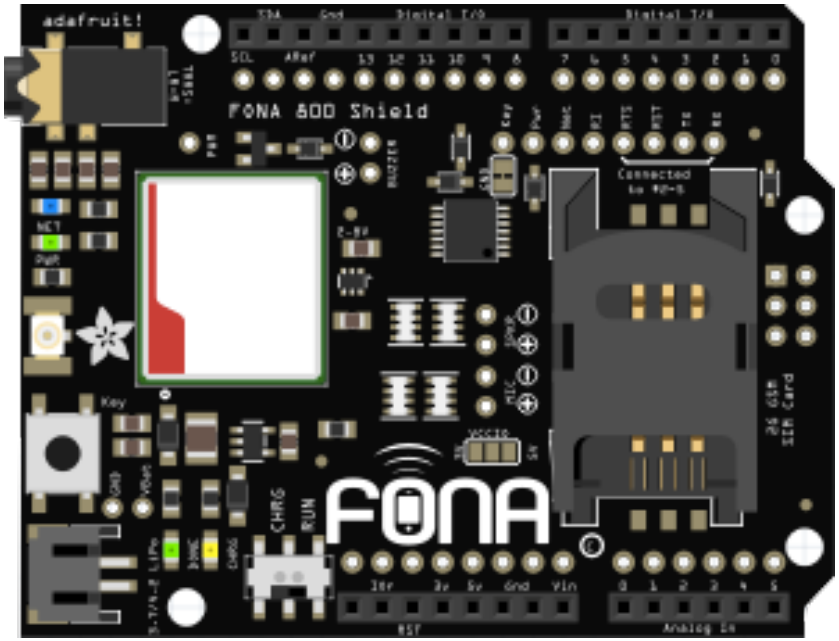
PHASE 1: THE APP

- Tools: Xcode, Swift, dweet.io
- UI: A switch and reset button that would allow users to send on, off, and reset requests to the GSM module
- Implementation: The app communicates with dweet.io by utilizing its web based RESTful API. User input on the switches and buttons are linked to codes that carry out the respective GET requests which are then sent as a JSON response to be received by the GSM module



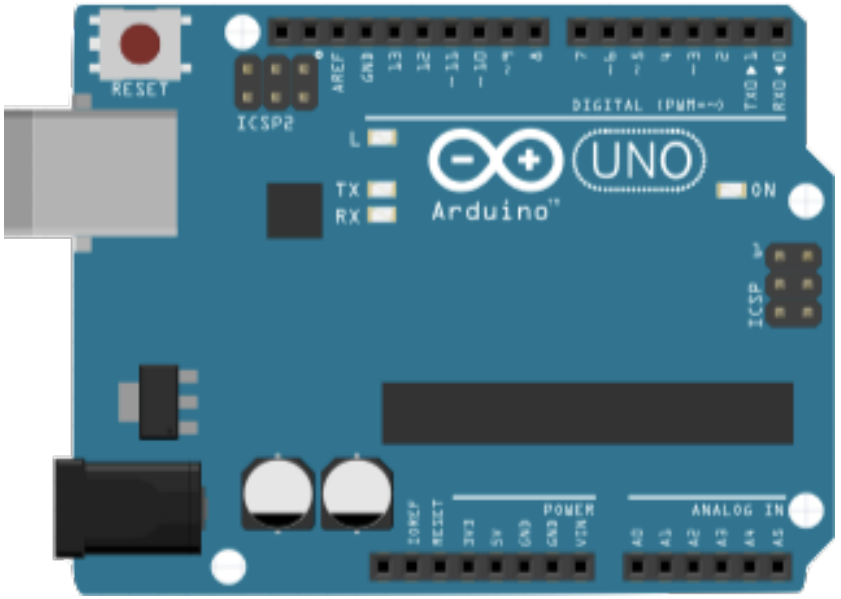
PHASE 2: THE GSM MODULE

- Tools: Adafruit FONA 800, Arduino IDE
- Assembly: The Adafruit Fona 800 GSM Module is fitted with an antennae and SIM card to carry out cellular communications. Pin outs are connected to the Arduino to relay incoming information from dweet.io to be read in the Arduino IDE
- Implementation: The Adafruit library was used to configure the backend connection of the GSM module while the incoming JSON response from dweet.io was to be read and recorded in the Arduino IDE's serial monitor

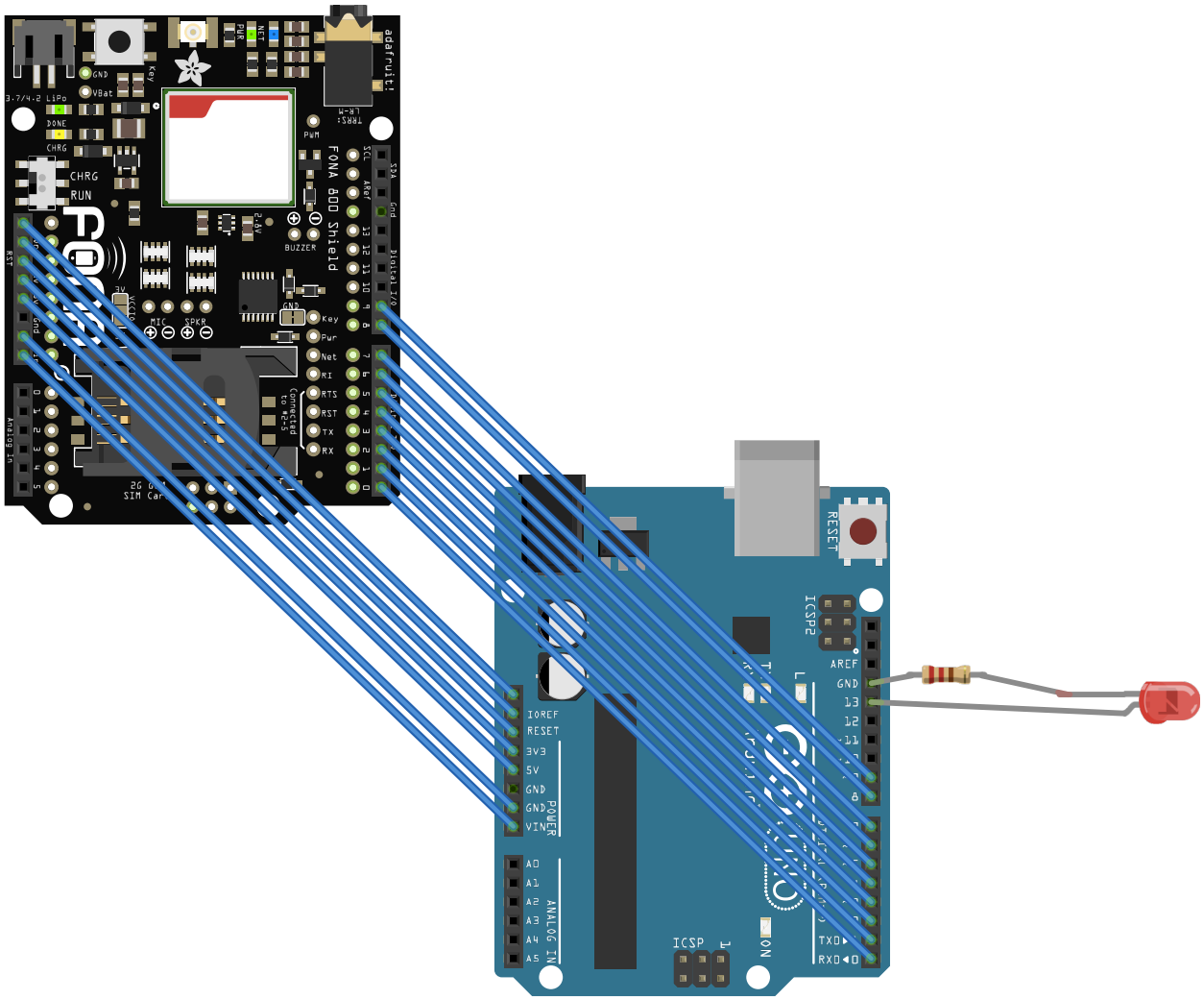


PHASE 3: THE ARDUINO

- Tools: Arduino IDE, Arduino UNO, LED
- Assembly: The Arduino Uno connects the information incoming from the GSM module to the micro-controller that carries out the on/off switching of the LED. The LED itself is placed in series with a resistor to a desired port on the micro-controller
- Implementation: Information relayed from the GSM module comes in the form of a JSON response which is parsed as a string in the serial monitor. If/else statements are put in place to detect the correct substring and the respective functionality that needs to be carried out is reflected in the state of the LED



GSM MODULE TO ARDUINO CONNECTION



FUTURE APPLICATIONS AND IMPROVEMENTS

- The project is a proof of concept that could be expanded to accommodate more device elements and relay information back and forth
- The app could be improved upon by allowing the user to tune device elements (ex. brightness)
- The UI itself can update when new devices are placed into the circuit, displaying information such as installed port

END

THANK YOU

```swift
//
//  ViewController.swift
//  Solaire
//
//

import UIKit
import Foundation

class ViewController: UIViewController {

    @IBOutlet weak var dweetText: UILabel!
    @IBOutlet weak var mySwitch: UISwitch!


    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
        mySwitch.addTarget(self, action: #selector(switchToggled(_:)), for:
         UIControlEvents.valueChanged)
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    @IBAction func dweet(_ sender: UIButton) {
        dweetText.text = "Reset"
        let url = URL(string: "https://dweet.io/dweet/for/board2server?led_1=reset")

        let task = URLSession.shared.dataTask(with: url!) { (data, response, error)
         in

            if let data = data {
                do {
                    // Convert the data to JSON
                    let jsonSerialized = try JSONSerialization.jsonObject(with:
                     data, options: []) as? [String : Any]

                    if let json = jsonSerialized, let url = json["url"], let
                     explanation = json["explanation"] {
                        print(url)
                        print(explanation)
                    }
                }  catch let error as NSError {
                    print(error.localizedDescription)
                }
            } else if let error = error {
                print(error.localizedDescription)
            }
        }
        task.resume()
    }
```

```
@IBAction func switchToggled(_ sender: UISwitch) {
    changeText()
}

func changeText() {
    if mySwitch.isOn{
        dweetText.text = "LED: ON"
        let url = URL(string: "https://dweet.io/dweet/for/board2server?
         led_1=on")

        let task = URLSession.shared.dataTask(with: url!) { (data, response,
         error) in

            if let data = data {
                do {
                    // Convert the data to JSON
                    let jsonSerialized = try JSONSerialization.jsonObject(with:
                     data, options: []) as? [String : Any]

                    if let json = jsonSerialized, let url = json["url"], let
                     explanation = json["explanation"] {
                        print(url)
                        print(explanation)
                    }
                }  catch let error as NSError {
                    print(error.localizedDescription)
                }
            } else if let error = error {
                print(error.localizedDescription)
            }
        }
        task.resume()
    }
    else{
        dweetText.text = "LED: OFF"
        let url = URL(string: "https://dweet.io/dweet/for/board2server?
         led_1=off")

        let task = URLSession.shared.dataTask(with: url!) { (data, response,
         error) in

            if let data = data {
                do {
                    // Convert the data to JSON
                    let jsonSerialized = try JSONSerialization.jsonObject(with:
                     data, options: []) as? [String : Any]

                    if let json = jsonSerialized, let url = json["url"], let
                     explanation = json["explanation"] {
                        print(url)
                        print(explanation)
                    }
                }  catch let error as NSError {
```

```
                    print(error.localizedDescription)
                }
            } else if let error = error {
                print(error.localizedDescription)
            }
        }
        task.resume()
    }
  }
}
```

```
// Adafruit FONA and Arduino Code

// Libraries
#include <Adafruit_SleepyDog.h>
#include "Adafruit_FONA.h"
#include <SoftwareSerial.h>

// LED pin
const int ledPin = 13;

// Pins
#define FONA_RX 2
#define FONA_TX 3
#define FONA_RST 4

// Buffer
char replybuffer[255];

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

// Fona instance
Adafruit_FONA fona = Adafruit_FONA(FONA_RST);
uint8_t type;

// Thing name
String yourThing1 = "board2server";
String yourThing2 = "server2board";

// RESET THESE: Flagging and String Checking
String incoming_stream = "";
bool gprs_flag = false;
bool dweet_flag = false;



void setup() {
  pinMode(13, OUTPUT);
  // Initial serial
  while (!Serial);
  Serial.begin(115200);
  Serial.println(F("FONA reading SMS"));
  Serial.println(F("Initializing....(May take 3 seconds)"));
```

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

type = fona.type();
Serial.println(F("FONA is OK"));
Serial.print(F("Found "));
switch (type) {
  case FONA800L:
    Serial.println(F("FONA 800L")); break;
  case FONA800H:
    Serial.println(F("FONA 800H")); break;
  case FONA808_V1:
    Serial.println(F("FONA 808 (v1)")); break;
  case FONA808_V2:
    Serial.println(F("FONA 808 (v2)")); break;
  case FONA3G_A:
    Serial.println(F("FONA 3G (American)")); break;
  case FONA3G_E:
    Serial.println(F("FONA 3G (European)")); break;
  default:
    Serial.println(F("???")); break;
}

// Print module IMEI number.
char imei[15] = {0}; // MUST use a 16 character buffer for IMEI!
uint8_t imeiLen = fona.getIMEI(imei);
if (imeiLen > 0) {
  Serial.print("Module IMEI: "); Serial.println(imei);
}

// Setup GPRS settings
//fona.setGPRSNetworkSettings(F("internet"));
while (!gprs_flag)
{
  fona.setGPRSNetworkSettings(F("wholesale"), F(""), F(""));
  delay (15000);
  if (!fona.enableGPRS(true))
  {
    Serial.println(F("GPRS re-establishing connection"));
  }
  else
  {
```

```
      Serial.println(" ∧____∧");
      Serial.println("(゜ヮ゜)");
      Serial.println("⊂     つ");
      Serial.println("(つ ノ");
      Serial.println("(ノ");
      Serial.println("GPRS connection successful");
      gprs_flag = true;
    }
  }
  delay(2000);
}

void loop() {
  // Loop
  // Prepare request
  uint16_t statuscode;
  int16_t length;
  String url = "http://dweet.io/get/latest/dweet/for/board2server_";
  char buf[80];
  url.toCharArray(buf, url.length());

  Serial.print("Request: ");
  Serial.println(buf);

  // Send URL to Dweet.io
  while (!dweet_flag)
  {
    if (!fona.HTTP_GET_start(buf, &statuscode, (uint16_t *)&length))
    {
      Serial.println("Dweet reattempt");
      dweet_flag = false;
      delay(2000);
    }
    else
    {
      dweet_flag = true;
    }
  }
  while (length > 0) {
    while (fona.available()) {
      char c = fona.read();
      incoming_stream += (char)c;
      // Serial.write is too slow, we'll write directly to Serial register!
#if defined(__AVR_ATmega328P__) || defined(__AVR_ATmega168__)
      loop_until_bit_is_set(UCSR0A, UDRE0); /* Wait until data register empty.*/
```

```
      UDR0 = c;
#else
      Serial.write(c);
#endif
      length--;
    }
  }
  fona.HTTP_GET_end();

  // PIN CONTROL
  if (incoming_stream.indexOf("\"led_1\":\"on\"") >= 0)
  {
    Serial.println("LED: ON");
    digitalWrite(13, HIGH);
  }
  else if (incoming_stream.indexOf("\"led_1\":\"off\"") >= 0)
  {
    Serial.println("LED: OFF");
    digitalWrite(13, LOW);
  }
  else
  {
    Serial.println("LED: RESET");
    digitalWrite(13, LOW);
  }


  // Reset flags for next run
  gprs_flag = false;
  dweet_flag = false;
  incoming_stream = "";
  delay(3000);
}
```