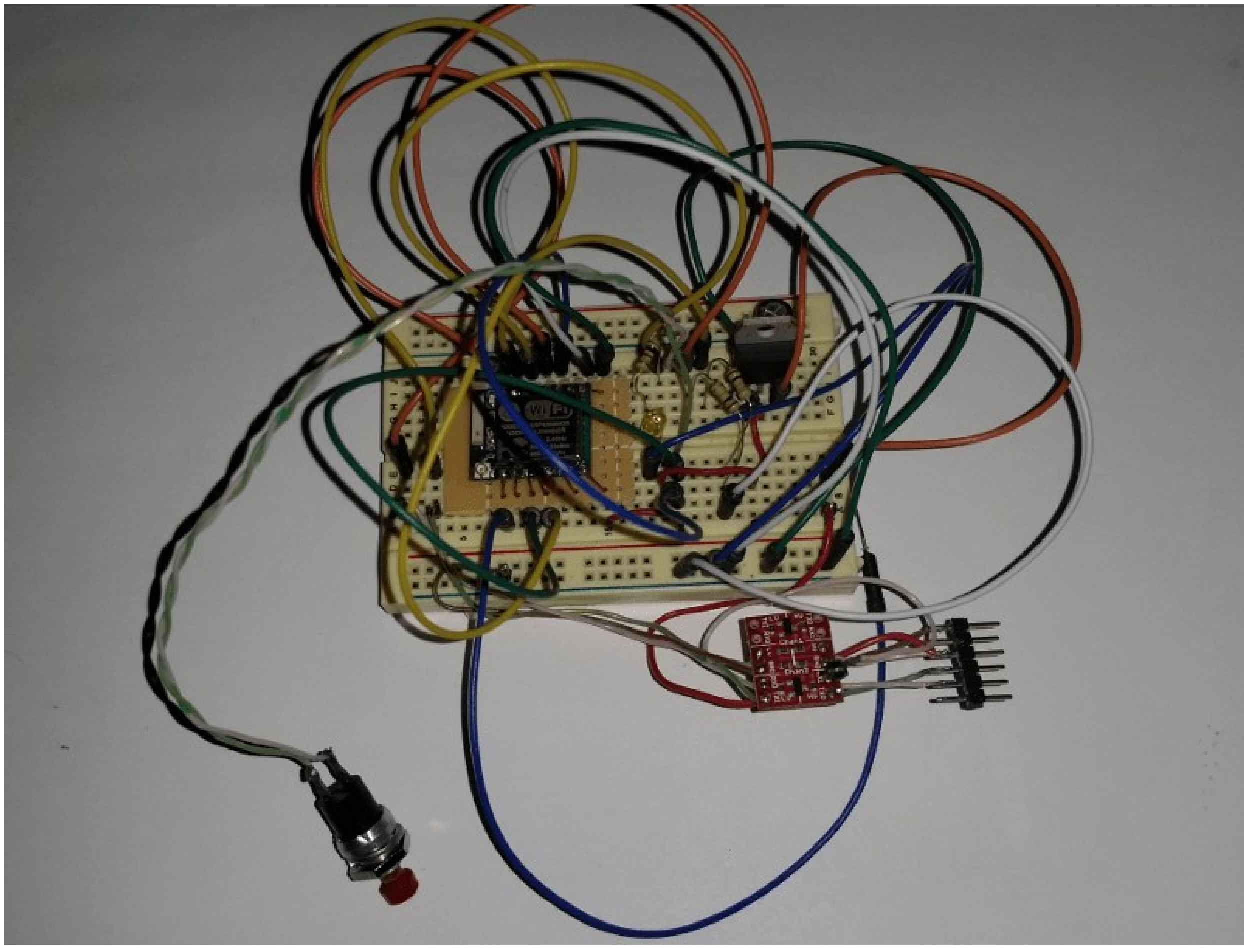
CULTURE / EDGE / IOT

Off-The-Shelf Hacker: Two-Way Comms with the ESP8266

2 Apr 2016 9:21am, by drtorq



1

With all the talk of the Internet of Things (IoT), the ESP8266 WiFi system-on-a-chip (SoC) is certainly a good candidate for those types of projects. It's fairly easy to program, has a small physical footprint and is inexpensive. IoT applications for the ESP8266 modules might include sensors, actuators, indicators and projects that need remote two-way communications.

communication between the remote node (the ESP8266 device) and my LAN, servers, the Cloud or user applications.

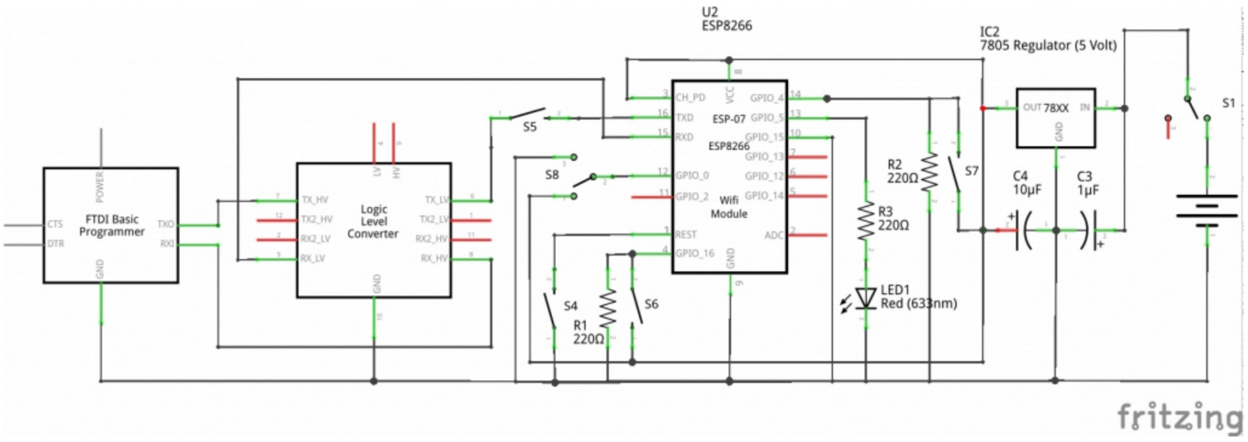
In this installment, I'll cover a basic communication scheme for the ESP8266-07 model. Feel free to expand the concepts to create your own IoT projects.

Frameworks and Setups

As we've discussed in earlier Off-The-Shelf Hacker stories, the ESP8266 chips are programmed using a variety of languages and frameworks. Personally, I like the Arduino IDE. The code syntax makes sense to me, the function libraries are pretty comprehensive and the environment is the same for 8266s, the Arduino and Processing project work. Be sure to review the previous "Read Schematics Like A Pro", "Add WiFi Connectivity to Your DIY Projects with the ESP8266" and "The Split Personality Of The ESP8266 WiFi Chip" The New Stack stories.

The setup for using an ESP8266-07 is similar to the -01 version, except that the -07 has an external antenna connector and more general purpose input/output (GPIO) pins. The physical break board looks pretty complicated.

The circuit is actually pretty straightforward, as shown in this schematic.



ESP8266-07 2-Way Comms Schematic

started with my 2-way communication routine.

The code is broken down into two parts, connecting to an access point (AP) and handling the GPIO.

Part 1 handles joining an access point and comes from the "AutoConnectWithFeedback" example program under the "WiFiManager" category. If the 8266 device hasn't connected to a particular access point before, it starts it's own access point and posts a page on its Web server (http://192.168.4.1), giving a user on a WiFi/browser equipped device (like a Galaxy 5 super phone) the option of choosing which AP to join. At this point, there is a temporary LAN between the 8266 device and the phone. The page then prompts the user for an AP access password. The 8266 will do a reset and connects to the chosen LAN while shutting down the temporary LAN and Web server. If the wifiManager.resetSettings(); line is commented out, it remembers AP credentials, so you don't have to enter the SSID and password each time. Uncomment the line to reset the settings, during programming and troubleshooting.

After the WiFi part of the code is initialized we move on to part 2.

Here's the code for part 1.

```
#include &lt;ESP8266WiFi.h&gt;
#include &lt;DNSServer.h&gt;
#include &lt;ESP8266WebServer.h&gt;
#include &lt;WiFiManager.h&gt;

void configModeCallback (WiFiManager *myWiFiManager) {
Serial.println("Entered config mode");
Serial.println(WiFi.softAPIP());
//if you used auto generated SSID, print it
Serial.println(myWiFiManager-&gt;getConfigPortalSSID());
}

const int ledPin = 5;
const int buttonPin4 = 4;
```

```
void setup(void) {
  Serial.begin(115200);

  WiFiManager wifiManager;
  wifiManager.resetSettings();
  wifiManager.setAPCallback(configModeCallback);

  if(!wifiManager.autoConnect()) {
    Serial.println("failed to connect and hit timeout");
    ESP.reset();
    delay(1000);
  }

  // Configure GPIO2 as OUTPUT.
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin4, INPUT);

  // Start TCP server.
  server.begin();
```

Manageable Code – Part 2

Part 2 handles reading the button, sending messages to remote machines (like my Galaxy superphone or Linux notebook) and optionally turning an LED on or off.

Interestingly, Part 2 just naturally breaks out in the looping part of the firmware, which actually does the work of the device. Additional functionality would most likely be inserted in this section.

Here's the code for Part 2.

```
void loop(void) {

  // Check if module is still connected to WiFi.
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("WiFi connected inside void loop");
    while (WiFi.status() != WL_CONNECTED) {
      Serial.println("WiFi.status connected loop");
      delay(500);
  WiFiClient client = server.available();

  if (client) {
    Serial.println("Client connected.");

    while (client.connected()) {

      buttonState4 = digitalRead(buttonPin4);
        // Serial.println(buttonState4);
        if (buttonState4 == HIGH) {
          // digitalWrite(5, HIGH);
          Serial.println("Button pushed");
          client.write("Button pushed\n");
        }

      if (client.available()) {

        char command = client.read();
        if (command == 'H') {
          digitalWrite(ledPin, HIGH);
          Serial.println("LED is now on.");
          client.write("LED is now on.");
        }
        else if (command == 'L') {
          digitalWrite(ledPin, LOW);
          Serial.println("LED is now off.");
          client.write("LED is now off.");
        }
     }
    }
    Serial.println("Client disconnected.");
    client.stop();
  }
}

void printWiFiStatus() {
  Serial.println("");
  Serial.print("Connected to ");
  // Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}
```

both a Linux notebook and my Galaxy 5 phone while assembling this story. When the button is pushed, the 8266 sends a "button pushed" text message to the remote client machine. If the remote machine sends an "H" character to the 8266 device, it lights up the LED. Likewise, sending an "L" character turns the LED off. Make your messages depending on your functional needs.

The printWiFiStatus() code simply sends status information to the serial terminal and is useful for development and troubleshooting.

Going Further

A basic two-way communication scheme is a reasonable starting point for expansion, with more buttons, sensors, outputs and so on. Since the LAN configuration code is built-in using library functions, it's also easy to use the device in different locations, with little extra effort. A little bit of planning and logistics could make the concepts outlined today scalable to larger projects with multiple devices and locations.

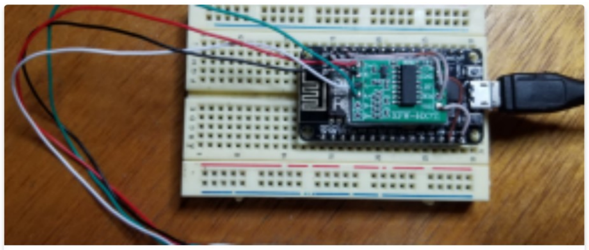
Once you have the buttons or sensors and output devices established, it's easy to change behavior by just modding and uploading new firmware.

You might look at the ESP8266WiFiMesh and ESP8266WiFi examples for additional project ideas.

OFF-THE-SHELF HACKER

1

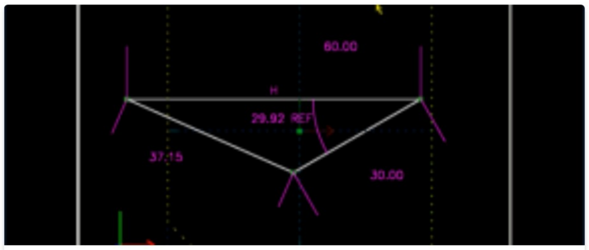
RELATED STORIES



EDGE / IOT

Off-The-Shelf Hacker: Feel Force with a Load Sensor

21 Sep 2019 6:00am, by drtorq



DEVELOPMENT / EDGE / IOT

Off-The-Shelf Hacker: Model Your Mechanisms with SolveSpace

14 Sep 2019 6:00am, by drtorq

View / Add Comments

Please stay on topic and be respectful of others. Review our Terms of Use.

SPONSORED FEED

harness

The Public Cloud can Teach Your Data Center New Tricks

SEPTEMBER 24, 2019

6 Things You Missed If You Didn't Visit Redis Labs at Oracle OpenWorld
SEPTEMBER 24, 2019

Meet the VMware Cloud Native Apps Field Engineering and Education Team
SEPTEMBER 24, 2019

AI Summit San Francisco: NetApp, Core Scientific, and OmniSci Showcase Cloud-Style NVIDIA-Powered AI
SEPTEMBER 24, 2019

Websockets is now part of the Tidelift Subscription
SEPTEMBER 24, 2019

OpenShift Scale-CI: Part 2 – Deep Dive
SEPTEMBER 24, 2019

Why we're reducing the time to payout and launching a bug bounty anniversary contest
SEPTEMBER 23, 2019

5 Reasons to Attend Accelerate Vienna: Automation @ the Speed of Change
SEPTEMBER 23, 2019

Microservice Security and Compliance in Highly Regulated Industries: Threat Modeling
SEPTEMBER 23, 2019

Container Journal: "Harbor Container Registry Project Advances"
SEPTEMBER 23, 2019

Facebook, Uber, Twitter and Alibaba form Presto Foundation to Tackle Distributed Data Processing at Scale
SEPTEMBER 23, 2019

The Hoot - Episode 7 - Index-free Logging With Greg Burd
SEPTEMBER 22, 2019

SEMAPHORE
SEPTEMBER 20, 2019

dynatrace
Deploying AI powered software intelligence in environments with technical or regulatory constraints
SEPTEMBER 20, 2019

influxdata
Monitoring & Alerting in InfluxDB Cloud 2.0
SEPTEMBER 20, 2019

HashiCorp
HashiCorp Consul at Cloud Field Day
SEPTEMBER 19, 2019

New Relic.
FutureStack19: Lew Cirne on Building the Industry's First Observability Platform
SEPTEMBER 19, 2019

circleci
Why CircleCI is a Leader in The Forrester Wave™: Cloud-Native Continuous Integration Tools, Q3 2019: speed, scale, security, and compliance
SEPTEMBER 19, 2019

puppet
Automation and parenting: data-driven diaper changing and more.
SEPTEMBER 19, 2019

epsagon
Tutorial: Building & Monitoring Serverless Web Applications
SEPTEMBER 19, 2019

CAPSULE8
Major Key Alert: Data Discovery for Red Teams with an ML Tool for Keylogging
SEPTEMBER 18, 2019

EXOSCALE
Fast Logs: Optimize Logging for Large Architectures
SEPTEMBER 17, 2019

RAYGUN
Make fewer HTTP requests: What this means and how to do it
SEPTEMBER 17, 2019

ATOMIST — Exploring the world with Aspects
SEPTEMBER 11, 2019

CLOUD FOUNDRY — Developers Across Europe Descend on The Hague for 2019 Cloud Foundry Europe Summit
SEPTEMBER 11, 2019

ORACLE — Extending Aqua Security to Oracle Container Engine for Kubernetes
SEPTEMBER 09, 2019

Lightbend — Full Stack Reactive In Practice [Webinar]
SEPTEMBER 09, 2019

HAPROXY — Introducing HAProxyConf 2019
SEPTEMBER 09, 2019

nirmata — Virtual Kubernetes Clusters
AUGUST 25, 2019

NS1. — Driving DNS with Network Latency Feedback at Salesforce
AUGUST 06, 2019

packet — Oops, We Forgot to Build a Managed Kubernetes Service!
MAY 03, 2019

THENEWSTACK Ebooks Podcasts Events Newsletter

Architecture Development Operations