Christoph Ruepprich
September 20, 2016

# BLE Beacon Gate Opener

## ODTUG GeekAThon 2016

### Team: Texas Heat

Christoph Alexander Ruepprich, a.k.a Rocket, Christoph Martin Ruepprich

### Concept

The BLE (Bluetooth Low Energy) beacon from the KScope 16 conference badge, located inside a car, is used to automatically open a driveway swing gate.

### Description

A battery powered Raspberry Pi 2B running Node.js is used to detect the beacon's bluetooth signal as the car passes by. When the signal is detected, the node.js program checks the identity of the beacon signal. If it matches the KScope 16 beacon's identity, a signal is sent to a port on the Raspberry Pi's GPIO pins. This pin, along with the 5.5 volt power and ground, is connected to a relay switch. The other side of the relay switch is connected to the momentary switch of the swing gate opener (the same as a garage door opener) remote control. When the relay is switched on, the circuit of the gate opener remote switch is closed, simulating the remote's button press, and the swing gate moves.

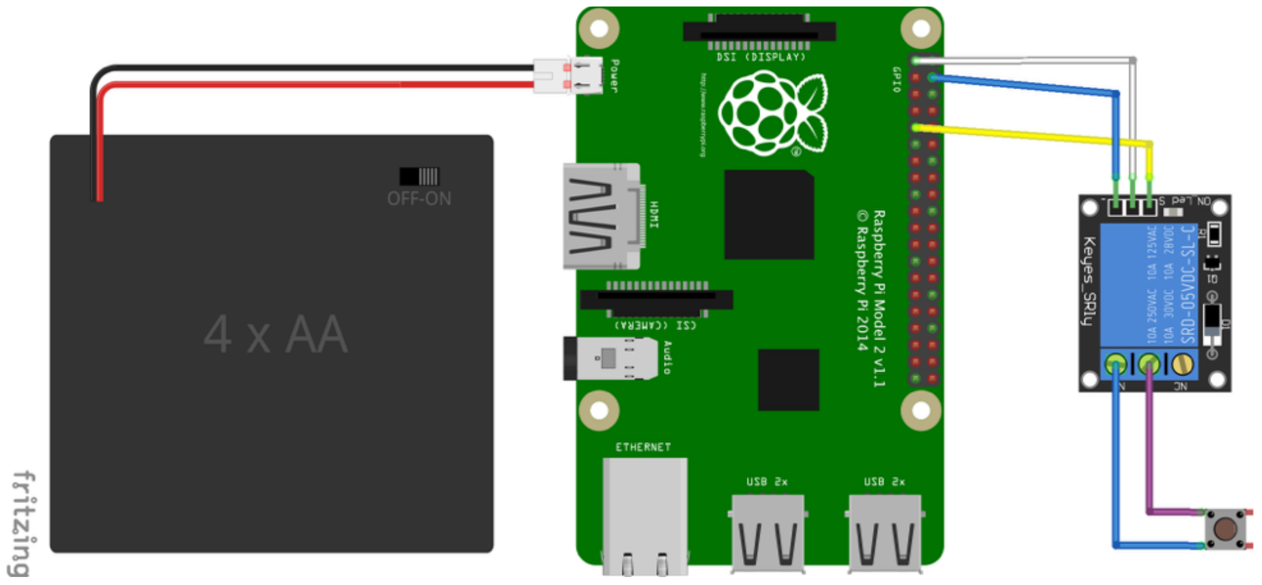
### Parts Used

- 1 Raspberry Pi 2 Model B (other models will probably work, too)
- 1 4 x AA battery pack with micro USB connector (https://amzn.com/B00BR22W0Q)
- 1 Keyes Relay Module (https://amzn.com/B00VRUAHLE)
- 1 Bluetooth USB Adapter (https://amzn.com/B009ZIILLI)
- Swing Gate opener remote control (the one I already had)
- Colored Wires with female pin connectors for Raspberry Pi

## Tools

- Small screw driver for relay module contacts
- Soldering Iron with solder

## Wiring Diagram



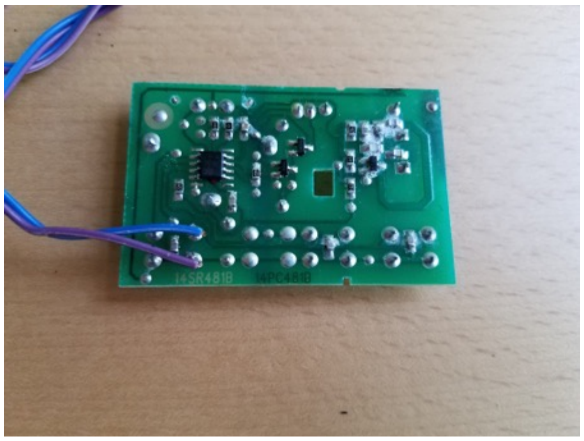
The center component shows the Raspberry Pi 2B. To its left is a battery pack with four AA batteries supplying power via the micro USB port on the Raspberry Pi. The blue component on the right shows the relay switch connected to a momentary switch (the button of the gate opener remote control).

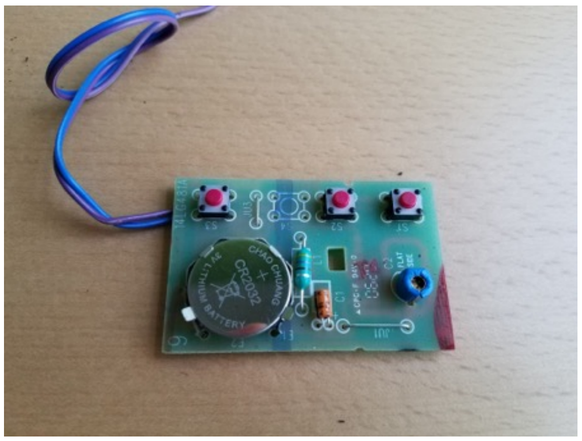
The wiring to from the GPIO pins to the relay:
White: 5v
Blue: Ground
Yellow: GPIO pin 4.

The wiring to the momentary switch (order does not matter):
Blue: switch lead 1
Red: switch lead 2

## Remote Control

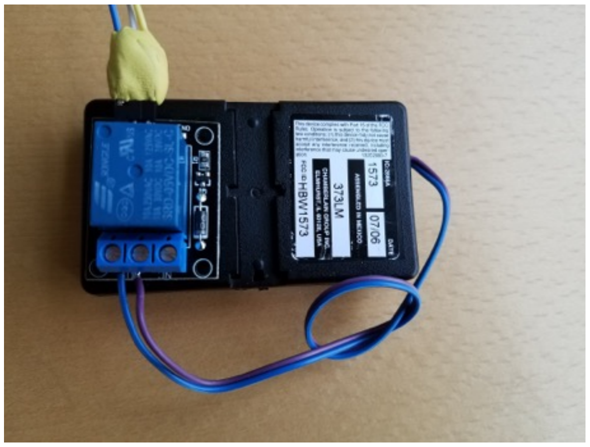
I removed the circuit board of the remote control in order to solder wires to the opposing leads of one of the remote control's buttons. I chose the button that was previously programmed to operate the swing gate.


Back of remote control circuit board with wires soldered to the momentary switch


Front of remote control circuit board, showing three switches and battery.

To test the correct wiring, I connected the ends of the wires, thus short circuiting the switch, which then operated the swing gate.

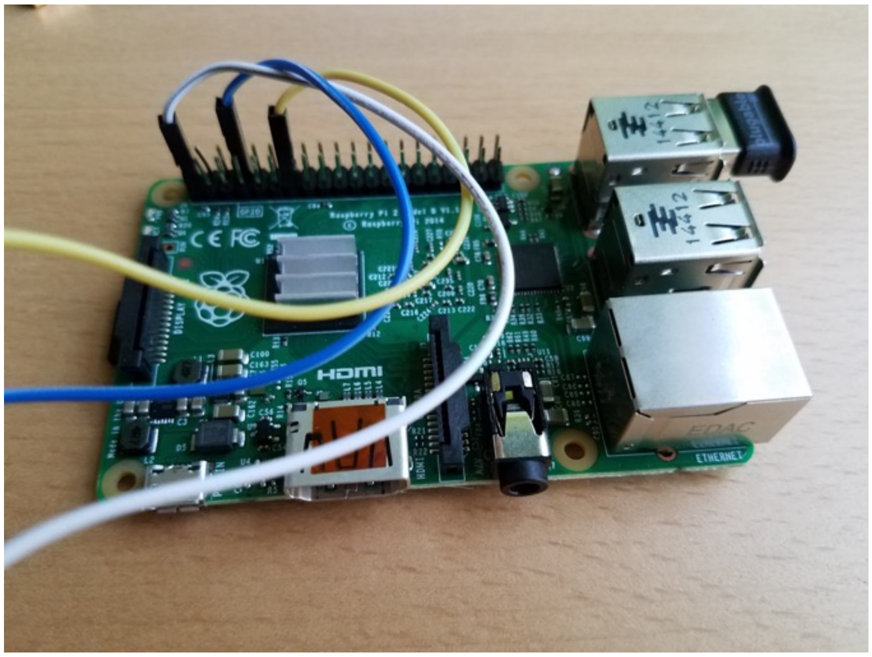
After replacing the circuit board in its case, I connected the two wires to the relay module.


Back of remote with the case replaced, the relay module and wires attached.

I attached the relay module with tack putty to the back of the remote control case. The wires leading to the GPIO are protected with more putty.

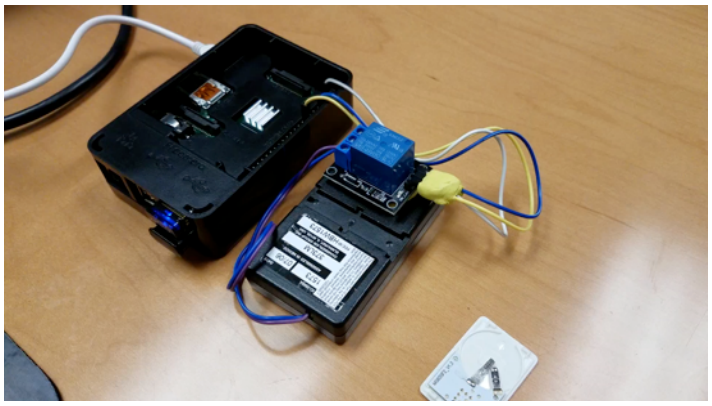
## Raspberry Pi

The image shows the Raspberry Pi with the wires leading from the GIPO pins to the relay module.



## Completed Setup

Here is the complete setup showing the Raspberry Pi in its case, the remote control with the relay module stuck to it, and the BLE beacon (open without battery).



## The Code

For this project I chose Node.js as the programming language. It's event driven nature lends itself nicely to an application that needs to listen for incoming signals. Being able to easily create a web server in Node.js and having and Oracle database driver available, I can further expand this project later to include a web interface and database interaction. For this project however, no web server was used, and no Oracle database interaction was needed.

I won't go into the details about installing Node.js. The installation is quite easy and can done through the official web site: https://nodejs.org/.

The application *gate.js*, which is the one used to read the beacon and triggers the relay, uses two node.js libraries that can easily be installed with the node package manager (npm). You can install these libraries either globally or locally into the directory where *gate.js* is located.

```
$ npm install onoff
$ npm install noble
```

The *onoff* library provides access to the GPIO pins. The *noble* library connects to the bluetooth USB dongle.

The gate.js program first sets up GPIO pin 17, to which I wired the yellow lead from the relay module.

Then the program enables bluetooth scanning with the *noble.on('discover'…)* method.

The *noble.on('discover'…)* method uses a callback function to identify the bluetooth signal received. If the ID of the bluetooth signal matches that of my beacon, the relay is activated with the *activateGate()* method. The *activateGate()* method will wait for seven seconds until the gate is fully open. This is necessary because the *discover* method with continuously see the beacon while it is in range, and attempt to continuously activate the gate. The seven second timeout will allow the gate to fully open, and the car to get out of bluetooth range before the relay can be triggered again.

## Videos

Swing gate opener in action: https://youtu.be/uE9ZzU-_gK8
Thanks to Christoph Ruepprich, Jr. for helping out with shooting video, testing and demoing.

Demo of BLE beacon triggering the relay module mounted on top of the garage door opener remote: https://youtu.be/8dMQKuyFzGc

# Appendix

## gate.js

```
/**
 * Node.js BLE enabled relay switch
 * Created for ODTUG Geekathon 2016
 * Source (GitHub): https://github.com/cruepprich/gateOpener/blob/master/gate.js
 * License: GNU General Public License, version 3 (GPLv3)
 *  - http://opensource.org/licenses/gpl-3.0.html
 * @author Christoph Ruepprich https://ruepprich.wordpress.com
 *
 * Used to read a bluetooth low energy emitter (BLE) beacon and trigger a swing gate
 * opener when the beacon comes into range.
 * Requires a Raspberry Pi 2 with a bluetooth USB adapter:
 * (https://amzn.com/B009ZIILLI)
 *
 * The noble library is used to read bluetooth signals.
 * (https://github.com/sandeepmistry/noble)
 *
 * The onoff library is used to access the GPIO ports on the Raspberry Pi.
 * (https://github.com/fivdi/onoff)
 **/


var RSSI_THRESHOLD    = -100;
var EXIT_GRACE_PERIOD = 2000; // milliseconds

var inRange = [];

var gpio = require('onoff').Gpio  //lib to address GPIO pins
  , noble = require('noble')      //lib to read bluetooth signals
  , remote = new gpio(17,'out')
  , gateState = 'STOPPED'         //captue gate state: MOVING,STOPPED
  ;


//Sends signal to switch the relay on for one second.
//This in effect simulates pressing the button on the
//garage door opener remote.
function pressButton() {
  //press button for n seconds
  remote.writeSync(1);
  setTimeout(function() {
    remote.writeSync(0);
  }, 1000);

}


//Set the gateState to MOVING. This prevents the gate
//being acivated when already moving.
//The gate moves about 7 seconds. After that time the
//state is set back to STOPPED so that the gate can be
//activated again.
function activateGate() {
  gateState = 'MOVING';
  pressButton();
  console.log('Gate state',gateState);
  //wait until gate is open
  setTimeout(function() {
    gateState = 'STOPPED';
  }, 7000);
}


//Main noble listener. When a BLE is detected we check if it is
//within an acceptable range. The we check whether we have detected
//it previously. If not, we check if it is our beacon (TurnoutNow)
```

```
//and activate the gate.
noble.on('discover', function(peripheral) {
  if (peripheral.rssi < RSSI_THRESHOLD) {
    // ignore
    console.log('ignoring peripheral.rssi',peripheral.rssi);
    return;
  }

  var id = peripheral.id;
  var entered = !inRange[id]; //checks if this id has been seen

  //if this is a new id, check if it is our beacon and activate the gate
  if (entered) {
    inRange[id] = {
      peripheral: peripheral
    };

    //our beacon has the localName TurnoutNow
    //if it is detected activate the gate
    if (peripheral.advertisement.localName == 'TurnoutNow') {
      console.log('"' + peripheral.advertisement.localName + '" entered (RSSI ' + peripheral.rssi + ') ' + new Date());

      //If the gate is not already moving, activate it
      if (gateState == 'STOPPED') {
        activateGate();
      }
    }
  }

  //record the current time of when this beacon has been seen
  inRange[id].lastSeen = Date.now();
});


//periodically check if the beacon is no longer of consequence
setInterval(function() {

  //loop through all the beacons that we saw recently
  for (var id in inRange) {

    //if we have not seen it for more than two seconds, we delete it
    if (inRange[id].lastSeen < (Date.now() - EXIT_GRACE_PERIOD)) {
      delete inRange[id];
    }
  }
}, EXIT_GRACE_PERIOD / 2);


//Check bluetooth state. If powered on then start scanning
noble.on('stateChange', function(state) {
  if (state === 'poweredOn') {
    noble.startScanning([], true);
  } else {
    noble.stopScanning();
  }
});
```