Garage Door Controller
Project Overview Document

By: Jim Schrempp and Bob Glicksman; v2, 7/20/2020

NOTICE: Use of this document is subject to the terms of use described in the document "Terms_of_Use_License_and_Disclaimer" that is included in this release package. This document can be found at:

https://github.com/TeamPracticalProjects/Garage_Door_Controller/Terms_of_Use_License_and_Disclaimer.pdf

TABLE OF CONTENTS.

OVERVIEW.

This document presents an overview of the *Garage Door Controller* project. This section outlines what the project is about. Section 2 describes the layout and contents of this repository – where files are located and what each file is about. Section 3 of this document provides a high level description of the theory of operation and the various components of the project.

The purpose of the *Garage Door Controller* is to allow a user to open and close their garage door using an App on their (Android[1]) smartphone. The project also provides the status of the garage door (open or closed) and this status is displayed on the App. Connection between the app and the controller hardware is via the Internet. Therefore, the App can control and status the garage door from anywhere Internet is available.

In order to provide Internet communication, the *Garage Door Controller* needs WiFi access to the Internet. WiFi access is provided via a Particle[2] Photon microcontroller module. In order to provide contactless sensing of the garage door position (open or closed), the *Garage Door Controller* needs to be located in the garage. Before embarking on this project, you must make sure that:

- There is a good quality WiFi signal where the *Garage Door Controller* hardware is to be placed;
- The location in the garage where the hardware is to be placed does not exhibit ambient temperatures below -4 degrees C (25 degrees F) or ambient temperature highs above 60 degrees C (140 degrees F); and
- There is an AC power source within about 10 feet of where the *Garage Door Controller* hardware is to be placed.

This project is released under an open source, non-commercial license. In order to use the materials in this repository, you must read and agree to all of the terms and conditions in the following document:

https://github.com/TeamPracticalProjects/Garage_Door_Controller/blob/master/Terms_of_Use_License_and_Disclaimer.pdf

This is an open-source project. This repository contains complete source files for all hardware and software. It also contains documents describing how to build this project, how to install this project, and how to use this project.

[1] At this time, the App is only available for Android devices. Our apologies to iOS (iPhone) users.
[2] https://www.particle.io

REPOSITORY STRUCTURE AND CONTENTS.

This section describes the contents of this repository. This is an open source project. The files in this repository contain source code, (references to) hardware schematics and CAD files, and documentation.

Repository Structure.

GitHub.com/TeamPracticalProjects/Garage_Door_Controller

- "*README.md"*: the overview read-me file that is displayed on the GitHub project page. The file format is *markdown* (.md)
- "*Terms_of_Use_License_and_Disclaimer*": the terms of use that you must agree to in order to use the contents of this repository. The file format is *portable document format* (.pdf)
- "*Overview*": This document. The file format is *portable document format*(.pdf)
- "*.gitattributes*": automatically provided by GitHub. You do not need to access this file. The file format is *text*.
- **Software** (folder containing all software and firmware):
  - **App** (folder containing the app software)
    - **executable** (folder)
      - "*GarageDoorController.apk*": Android install package for the app. Load this file onto your Android smartphone and tap on it to install the app. The file format is *android application package* (.apk)
    - **src** (folder)
      - "*GarageDoorController.aia*": Source code for the app. The source code is in the MIT App Inventor 2 graphical language. Import this file into your MIT App Inventor 2 IDE in order to view and/or edit the app source code. The file format is *AI2 source code format* (.aia).
  - **Photon** (folder containing the firmware for the Photon)
    - **src** (folder)
      - "*GarageDoorController.ino*": Particle Photon source code for the firmware that is loaded onto the Photon. The file format is *Arduino source code format* (.ino), which is pure text. Copy the contents of this file into the Particle Web IDE or the Particle Workbench in order to compile the firmware and flash it onto your Photon.
- **Docs** (folder containing project documentation)
  - "*Garage_Door_Controller_Build_And_Installation_Instructions*": Document that details how to build, install and test the project. The file format is portable document format (.pdf).

| -- "*Garage_Door_Controller_User_Manual*": User manual for the App. The file format is *portable document format* (.pdf).

Suggested Use of the Documents.

We suggest that you read the documents (.pdf files) contained in this repository in the following order:

- *Terms_of_Use_License_and_Disclaimer*: read and agree to everything in this document before proceeding further this this project.

- *Overview*: read through this document to familiarize yourself with what this project is about and ensure that you can successfully complete, install and use this project before you invest further in it.

- *Garage_Door_Controller_Build_And_Installation_Instructions*: follow the steps in this document to build the electronics, package the electronics, install the firmware, install the App software, mount the hardware in your garage, and test that the project is working.

- *Garage_Door_Controller_User_Manual*: consult this manual to learn how to use the App.

OVERVIEW THEORY OF OPERATION.

The basic idea behind the *Garage Door Controller* is to not interfere with the garage door mechanism itself. Rather, a remote garage door opener is used that is already paired with the garage door unit. The remote device is modified by placing a set of relay contacts across the pushbutton on the remote. Closing the relay simulates pressing the remote device's activation button.

A non-contact means of determining garage door open/closed status is to measure the distance between some location in the garage and the door; e.g. distance from the overhead rafters to a solid object in the path where the door opens. An HC-SR04 ultrasonic ranging unit is used to measure the distance and a suitable threshold is established in microcontroller firmware. In our case, an open garage door presents an obstacle at a little over 12 inches from the rafters. When the garage door is closed, it is no longer in the distance measuring path and the distance to the nearest obstacle (a car or other material) is much greater (in our case, distance to a car parked under the ultrasonic unit is at least 42 inches and the distance to the floor, when the car is not parked in the garage, is over 99 inches).

Figure 3-1 shows the overall system architecture of the project:

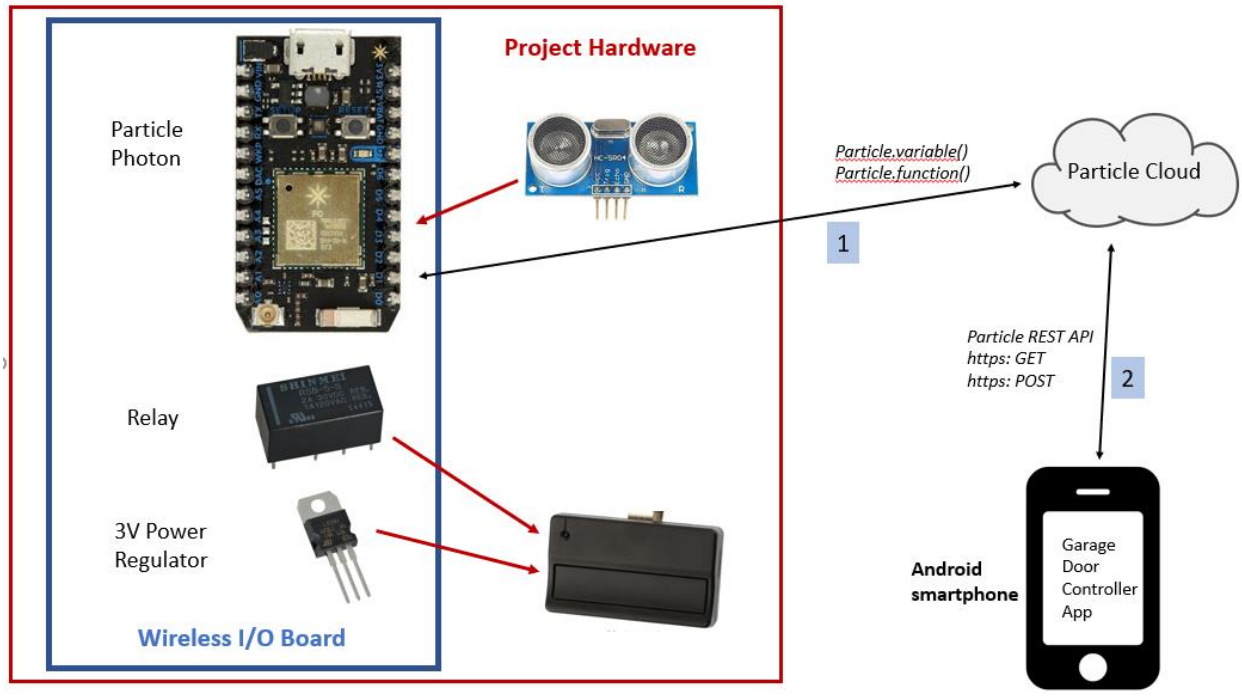

Figure 3-1. System Architecture.

We used our Wireless I/O Board to provide the main controller electronics. For further details about the Wireless I/O Board, see:

https://github.com/TeamPracticalProjects/Wireless_IO_Board

The Wireless I/O Board houses a Particle Photon. The Photon is a powerful, yet inexpensive microcontroller module with built in Wifi and full IOT[3] support via the Particle cloud. See https://www.particle.io for further details about Particle, the Photon hardware, and the Particle IOT cloud infrastructure.

The Wireless I/O Board also houses a relay and associated drive electronics. The relay is controlled in firmware via Photon pin D0. The relay contacts are accessible via a 3 position terminal block on the board. The Wireless I/O Board has provision for mounting a 3.3 volt power regulator (different from the Photon's on-board 3.3 volt regulator) and this power supply can be used in lieu of the remote device's battery. This feature is particularly handy when the electronics are located up in the garage's rafters and replacing batteries would be inconvenient at best.

The Wireless I/O Board has a 3-pin header for connecting a hobby servo. This project does not use a servo. The servo control is via Photon pin D1. The signal from the Photon is amplified to 5 volts, which is suitable for triggering the HC-SR04 sensor. The 5 volt Vcc power and ground for the HC-SR04 is also available on this pin header. Photon pin D4 is directly connected to a sensor input block on the Wireless I/O Board and we used this pin to connect the HC_SR04 "echo" signal back to the Photon as a digital input (this pin is 5 volt tolerant).

Referring to figure 3-1: two cloud variables (Particle.variable()) are defined in the Photon firmware to hold the value of the distance measurement by the ultrasonic sensor and the open/close decision made by the firmware based upon this distance. The firmware simply takes a distance measurement every one second and stores the latest values in these global variables. Particle's IOT infrastructure allows Internet connected devices to access these two variables via the Particle cloud – see communication path (1) in the figure. Particle's REST API is used by the Android App to read out these distance and status values upon demand of the user, see communication path (2) in the figure. The REST API uses an https: GET to retrieve the values from these cloud variables.

A firmware function running on the Photon is also made accessible to the Particle cloud (Particle.function()). This function can be triggered by using Particle's REST API using an https: POST. This cloud function, running on the Photon, simply activates the on-board relay for a specified period of time (we found that 2 seconds worked well with our particular remote opener device). This has the same effect as pressing the remote opener button for 2 seconds.

Particle's REST API is very secure and uses industry standard encryption and OAUTH2 authentication to ensure that only those with proper credentials can access Particle's devices. Further information about Particle's REST API can be found at:

[3] IOT = Internet of Things

https://docs.particle.io/reference/device-cloud/api/

We previously published a project that details how an MIT App Inventor 2 app can log-in to the Particle cloud, select a Particle device in the logged-in user's account for use by the app, and then call cloud functions on that Particle device and read cloud variable data from that device. For details, see:

https://github.com/TeamPracticalProjects/MIT-App-Inventor-Particle-Photon-test

A note on the garage door opener: our project integrates with a Liftmaster garage door opener with a purple "learn button". A compatible remote control device can be purchased at:

https://smile.amazon.com/gp/product/B075MQCH2P/ref=ppx_yo_dt_b_asin_title_o02_s00?ie=UTF8&psc=1

This particular model device is compatible with many different Liftmaster and Chambertin garage door openers. However, you should be able to hack any remote device that is compatible with your garage door opener system.

Figure 3-2 shows how we "hacked" this particular garage opener remote:

Figure 3-2. Remote Control "Hack".

We found that the pushbutton (clearly visible at the lower right of the photo) simply grounds the contact where we have soldered a wire. This wire goes to the normally open (NO) contact of the relay on the Wireless I/O Board. We found a convenient ground connection on the device – this is where we tacked on the black wire. The device ground connects to the power ground on the Wireless I/O Board, which we also wired to the COM contact of the relay. We removed the 3 volt Lithium battery from the unit and tacked a wire to the top of the battery connector which is where the battery "+" terminal contacts. We ran this wire to the +3.3 volt power regulator output from the Wireless I/O Board.

This “hack” of the remote device is simple and straightforward. You should have an easy time doing this, even if your compatible garage door remote differs from ours. Further instructions can be found in the following document in this repository:

https://github.com/TeamPracticalProjects/Garage_Door_Controller/blob/master/Docs/Garage_Door_Controller_Build_And_Installation_Instructions.pdf

Our final installation is shown in the photo in figure 3-3, below:

Figure 3-3. Final Installation.

Note the Wireless I/O Board and modified remote control inside the plastic project box and the downward facing ultrasonic distance sensor below and to the right of the project box. A USB cable at the top of the project box brings 5 volt USB power from a USB “wall wart” to power the entire project.