Garage Door Opener Requirements

Mark Broihier
June 30, 2018

Table of Contents

1 Introduction

The purpose of this document is to define the requirements for an application that implements the remote control of a garage door opener.

1.1 Scope

This document defines the interface, processing, and displays for an application that receives a command from an Android device, authenticates it, and then sends a toggle command to an Overhead Door model 456 garage door opener that will command the door to either open or close depending on its current state. The requirements for both the Android device app and the garage door opener controller app are defined within this document. The collection of programs will be know as Garage Door Opener (GDO).

1.2 Features

GDO will consist of two applications running on two programable devices. One device will have a user interface that allows the user to command a garage door to open or close. The second device will accept the command from the first device and actuate a switch connected to an Overhead Door model 456 garage door opener.

1.3 Acronyms

GDO	Garage Door Opener
GPIO	General Purpose I/O
I/O	Input/Output

2 General Overview

The GDO Android App will have a simple, single button screen. When the button is pressed, a message will be sent by bluetooth to another device implementing a bluetooth server that will authenticate the message and then actuate a switch that will act as a push button to an Overhead Door model 456. Closing this switch will open a closed garage door or close an open garage door. The door opener's safety mechanism will be kept fully in tact, so if obstacles are detected in the path of the door, the activation of the switch will have no affect.

3 GDO

The following paragraphs outline the operation of GDO and define the requirements of it's behavior.

3.1 Theory of Operation

A Raspberry PI 0 W will host a portion of the GDO application. It will have the bluetooth device that will act as a server to accept remote commands and the general purpose I/O (GPIO) pins that will set the signal levels that will open or close the switch

connected to the Overhead Door model 456.

When powered on, this device will turn on the bluetooth transmitter / receiver and start a server to handle bluetooth messages between it and a paired device. It will also initialize the GPIO such that the switch is commanded open.

When the PI 0 W receives a command from a paired bluetooth device, it will authenticate it and send a reply of either accepted or rejected. If the command was accepted, the switch will be pulsed by sending a high (3.3 volts) command to a GPIO pin connected to a switch and then a low (0.0 volts).

An Android device will implement the user interface which will consist of a simple button and a status field. When the button is pressed, the device will send an encoded message to the PI 0 W via bluetooth. The interface will display the status received from the controller. The status will be: successful request, authentication failure, or communication failure. If the status from the PI 0 W was successful, it will mean the PI attempted to command the door open or closed. If the status was authentication failure, it means the encoded message was considered to be from an invalid source. If the status was communication failure, it means the message send / receive path was incomplete.

3.2 External Interface Requirements

ID::GDO-UI-EXT-IF-010 The GDO user interface shall have one button that will be used to command the door open or closed.

ID::GDO-UI-EXT-IF-020 The GDO user interface shall display a momentary status message after the button is pressed that shall display the status of the command attempt.

ID::GDO-UI-EXT-IF-020.001 The GDO user interface shall display a successful message if it receives a successful reply from the Raspberry PI 0 W.

ID::GDO-UI-EXT-IF-020.002 The GDO user interface shall display an authentication failure message if it receives an authentication failure response from the Raspberry PI 0 W.

ID::GDO-UI-EXT-IF-020.003 The GDO user interface shall display a communication failure message if it receives no response from the Raspberry PI 0 W within five seconds.

ID::GDO-PROC-EXT-IF-010 The GDO command processor (Raspberry PI 0 W) shall have one GPIO pin dedicated to actuating a switch connected to an Overhead Door model 456. This pin will be directly connected to the relay command pin.

ID::GDO-PROC-EXT-IF-020 The GDO command processor (Raspberry PI 0 W) shall have one GPIO pin dedicated to supplying power to a switch connected to an

Overhead Door model 456. This pin will be directly connected to the relay power pin.

ID::GDO-PROC-EXT-IF-030 The GDO command processor (Raspberry PI 0 W) shall have a ground GPIO pin directly connected to the relay ground/common.

ID::GDO-RELAY-EXT-IF-010 A relay channel associated with the actuating control of GDO-PROC-EXT-IF-010 shall have one terminal connected to the Overhead Door electrical connector identified as “PUSHBUTTON”.

ID::GDO-RELAY-EXT-IF-020 A relay channel associated with the actuating control of GDO-PROC-EXT-IF-010 shall have another terminal connected to the Overhead Door electrical connector identified as “COMMON”.

3.3 Processing Requirements

The following paragraphs identify the processing to be performed by the GDO Android User Interface and the Raspberry PI door controller device.

3.3.1 User Interface

ID::GDO-PR-UI-010 The user interface shall be implemented on an Android Device that supports bluetooth.

ID::GDO-PR-UI-020 When the Android app is started a main page shall display as a minimum:

ID::GDO-PR-UI-020.001 A text field that displays the status:

ID::GDO-PR-UI-020.001.001 The app shall display a status of “Ready” when it is started.

ID::GDO-PR-UI-020.001.002 The app shall display a status of “Communication Failure” when it fails to send a command or receive a response from the controller.

ID::GDO-PR-UI-020.001.003 The app shall display a status of “Command Successful” when it receives a successful indication from the controller.

ID::GDO-PR-UI-020.001.004 The app shall display a status of “Authentication Failure” when it fails to send a message considered valid by the controller.

ID::GDO-PR-UI-020.002 Display a command button.

ID::GDO-PR-UI-030 When the command button is pressed, the app shall:

ID::GDO-PR-UI-030.001 Open a bluetooth connection to the controller.

ID::GDO-PR-UI-030.002 Send an authentication message to the controller.

ID::GDO-PR-UI-030.003 Wait for and accept a controller reply.

ID::GDO-PR-UI-030.003.001 The reply shall be displayed in the status field of the main menu.

ID::GDO-PR-UI-030.003.002 If the reply is not received within an acceptable time, a Communication Error status message shall be displayed.

ID::GDO-PR-UI-030.004 Close the bluetooth connection to the controller.

3.3.2 Garage Door Controller

ID::GDO-PR-CONTROLLER-010 After power application to the controller, the controller shall:

ID::GDO-PR-CONTROLLER-010.001 Initialize the push button connection to the garage door opener to open (no button pressed).

ID::GDO-PR-CONTROLLER-010.002 Start a bluetooth server.

ID::GDO-PR-CONTROLLER-020 The bluetooth server shall:

ID::GDO-PR-CONTROLLER-020.001 Accept a single client connection.

ID::GDO-PR-CONTROLLER-020.002 Accept a single byte stream from the client limiting the stream to a fixed maximum number of bytes.

ID::GDO-PR-CONTROLLER-020.003 If more bytes are available, that is considered an authentication failure.

ID::GDO-PR-CONTROLLER-020.004 Apply an algorithm to assure the bytes are expected and have not been observed before. This authenticates the client.

ID::GDO-PR-CONTROLLER-020.005 If the client authenticates, close the push button connection to the garage door for a fixed number of seconds.

ID::GDO-PR-CONTROLLER-020.006 Send a reply of Command Successful if authentication was successful and Authentication Failure otherwise.

ID::GDO-PR-CONTROLLER-020.007 Close the client connection.

3.4 Provisioning Requirements

ID::GDO-PROV-010 The bluetooth connection shall be manually provisioned via Java code in the Android App and/or Android OS.

ID::GDO-PROV-020 The bluetooth connection shall be manually provisioned via code in the controller program and/or Linux OS.

3.5 Performance Requirements

ID::GDO-PERF-010 From button press to status display update shall take no longer than 5 seconds.

ID::GDO-PERF-020 Upon power loss, the controller shall take no longer than 1 minute to recover (able to accept a bluetooth remote commands) once power is again available.

ID::GDO-PERF-030 If a power cycle occurs, the return of power itself shall leave the power relay open. This will prevent the door from opening simply due to a power cycle.