GNG1103/2101

Design Project User Manual

RELAY FOR BYTES

Submitted by:

Relay for Bytes, Team 8

Thomas Alkhoury, 300066794

James Hight, 300102218

Laura Godfrey, 300082954

Cecilia Lou, 300055620

Julian Ward, 300076084

December 6, 2019

University of Ottawa

Abstract

The user manual will instruct the user how to re-create a relay system for a machine. The relay system will include a combination of software that determines if one has training as well as hardware the intercepts the current to an outlet depending on the training. It will offer instruction on how to set up a user interface that will indicate if the machine has electricity supplied to it. It will instruct the user how to connect the Raspberry Pi with the Dashboard user interface and the relay. It will also instruct the user how to correctly put together all the hardware to produce a relay-controlled outlet. The user manual will also include a bill of materials and an equipment list for the user. This manual also instructs the users how to use the product, from the software side to the mechanical side. It will also describe to the user the proper way to maintain the prototype. This included how to conduct regular checks for the connections and how to maintain the physical soldered connections. Finally, this manual recommends future actions for the users. This includes the future use of the project as well as how to expand the scope of the project. It explores how this product can be adapted for different uses.

Table of Contents

List of Figures

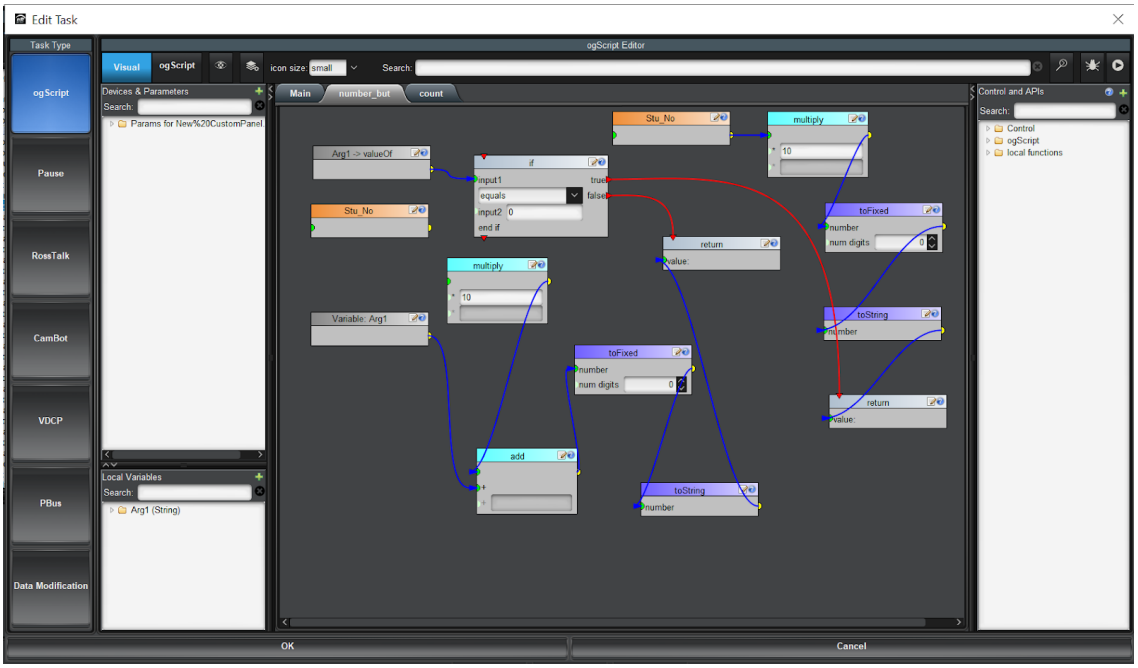

Figure 1: Dashboard Code for Number Buttons

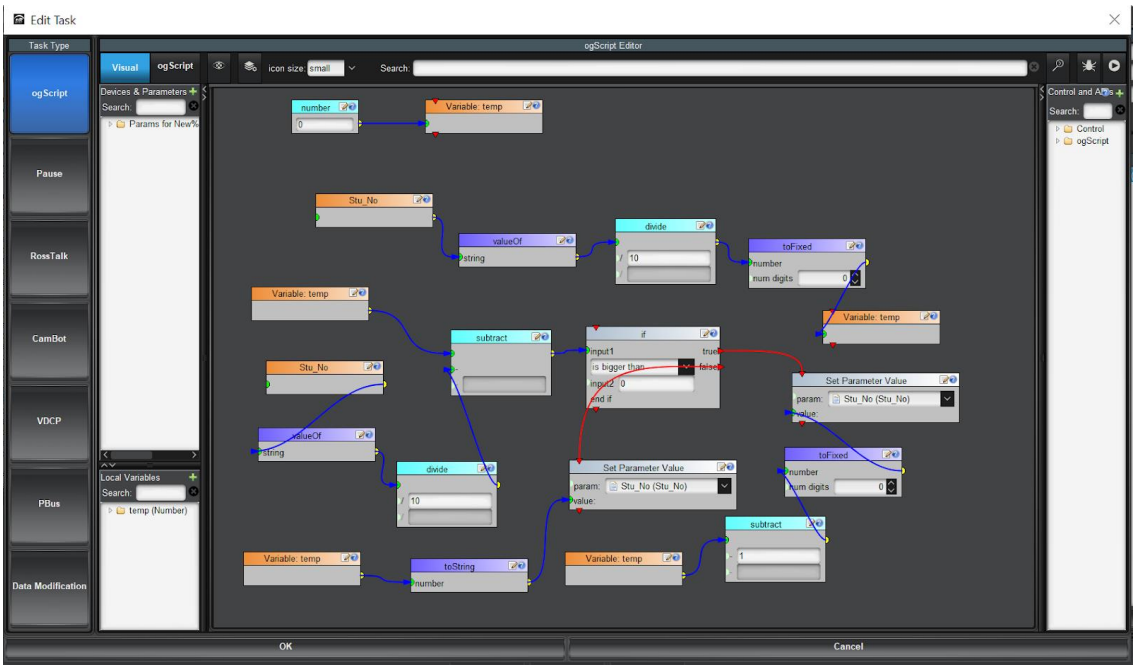

Figure 2: Dashboard Code for Backspace Button

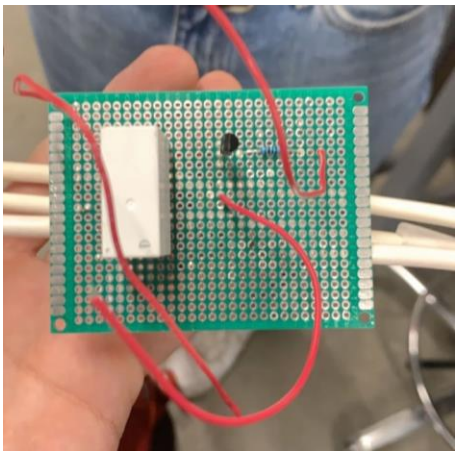

Figure 3: Top of PCB with Regular Relay

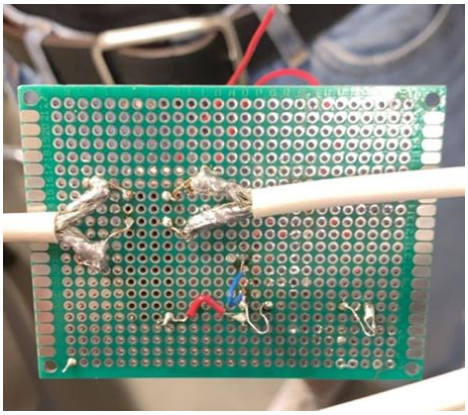

Figure 4: Underneath of PCB with Regular Relay

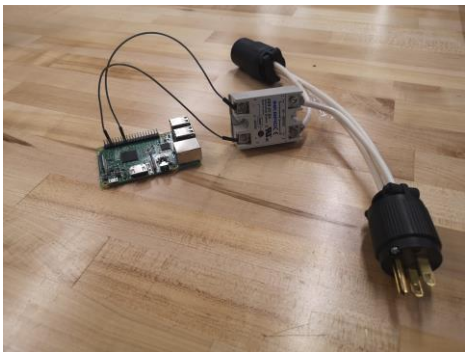

Figure 5: Solid State Relay with Raspberry Pi

List of Acronyms

Acronym	Definition
CEED	Centre for Entrepreneurship and Engineering Design
CSA	Canadian Standards Association
GND	Ground
GPIO	General Purpose Input/Output
PCB	Printed Circuit Board
RPi	Raspberry Pi 3
SSR	Solid State Relay
UI	User Interface

1 Introduction

1.1 Defining the Problem

This manual aims to outline and signify the work done to create a product, at the request of CEED that would assist in automating one aspect of their operations using Ross Video's Dashboard as a UI, making the work of its employees and clients easier. After meeting with the clients (Ross Video and CEED), and learning their needs and expectations, the problem that was set out to solve is that there are machines, especially in Brunsfield, that require training to use, but the only enforcement is a CEED staff at the door who manually checks the training. There must have been an easier, automated way to check the training.

1.2 Defining the Solution

A general solution to this problem can be shown in the block diagram below. There is no continuous input because, upon unsuccessfully trying to use the machine, the user will have to complete the training before trying again.

Ideally, this solution would require little to no input or upkeep from staff, which would allow them to attend to their other duties in Brunsfield, such as machine maintenance or assisting patrons with their projects. It can also help eliminate human error by automatically checking the user's credentials, instead of someone doing it manually and possibly getting a false positive or negative.

The final prototype implemented involves connecting Dashboard to a Raspberry Pi, which in turn controls a solid-state relay that controls current flow to the device. A user is able to enter their student number in the Dashboard interface, which sends a boolean signal to the RPi.

This product is the only one that addresses CEED's specific need of monitoring Brunsfield and regulating who uses the machines.

1.3 Constraints

The final prototype produced in this project has a couple of notable constraints. The largest one is likely the lack of CSA-approved equipment and safety regulations. Because the project involves live wires that carry a fairly large current (directly from a wall outlet), CEED management has said they would not be able to use Relay for Bytes in its current state, but future applications could use CSA-approved components safely.

The next largest constraint is cost. The current iteration of Relay for Bytes involves using one RPi unit connected to a monitor for each machine, which would quickly become unfeasible after expanding the system to include more machines. A potential workaround to this would be excluding Dashboard from future use, which would also drop the need for monitors.

1.4 Report Layout

This report will go through each subsystem of the devised solution and explain how it was built and designed. The aspects detailed in this manual are the Dashboard, Raspberry Pi, and the relay and wiring setup. A tutorial of how to use the final prototype and a maintenance guide are also included, which explain how best to safely handle the prototype and its components.

2 How the Prototype is Made

2.1 Software

2.1.1 Dashboard

Dashboard was crucial software that was required to be implemented into the product. Dashboard features an easy to manufacture coding process. Its visual coding process allows novice coders to easily implement a user interface for the product. It was also a much faster process than other user interface options. Its easily implemented buttons and other features allows for a really quick process when creating the user interface. Given the short timeline for the project, Dashboard was a great option for this project.

2.1.1.1 BOM (Bill of Materials)

- Dashboard Software

2.1.1.2 Equipment List

- Computer

2.1.1.3 Instructions

The first and most crucial step is to download the dashboard software from the Ross video platform. After the software has been downloaded and installed, the user interface can start to be created. First create a parameter and name its student number. Then create a grid of 3 by 4, in each square create a button. Each button in the top 3 by 3 is a number from one to 9. This is done through the code as seen in Figure 1 in the List of Figures. This is a basic function that can be applied to any

of the number buttons just by changing the input parameter. In the main code, the function is called and the number corresponding to the button is set as the value for Arg1 which refers to the input value. The backspace is created using a code as seen in Figure 2. The enter button is used to send the student number entered by the users to the RPi. This is done by using a function called ‘Send TCP String with Response.’ It sets the student number parameter as the string to be sent. The RPi’s pin and ip address as the pin and host. It sets the response end as ‘\t’. Finally, the callback function is

function callbackfunction (success, sentData, resultString, exception) { ogscript.debug("resultString is " + resultString);params.setValue('M1_Use',0,resultString) }.

This callback function displays the response of the RPi as indication as if Electricity is on or off by setting the value of the parameter display for Machine 1 to the response.

2.1.2 Raspberry Pi

The role of the RPi within this project is very important as it was the component telling all the other components what to do. The dashboard code sends a student number or information regarding the student’s training to the RPi. The RPi then takes this information checks it against a list of student numbers that do have the training and if this particular student number does have the training it sends a message to the relay to turn the electricity on. It also sends a reply to dashboard with a message saying, ‘Electricity On’. The opposite is true for a student number that does not have the training. If a student number does not have the required training associated with it then the RPi sends a message to the relay indicating to keep the electricity turned off. The RPi will then send a message to the Dashboard saying, ‘Electricity Off’. We considered using an Arduino board for

this project however the Raspberry Pi's computing capability seemed much more appropriate for what we wanted to accomplish as well the RPi will integrate best with CEED's current tap in system once that step is reached.

2.1.2.1 BOM (Bill of Materials)

- Raspberry Pi (model 3 was used for this project but any could be used as long as they are Wi-Fi-enabled)
- HDMI cord
- USB Keyboard
- USB Mouse
- Private Wi-Fi connection (or mobile hotspot from computer or phone)
- Power Cable for the Raspberry Pi
- Monitor
- Personal Computer

2.1.2.2 Equipment List

No equipment was needed to build the RPi. It comes as itself and all that's needed to use and operate the RPi is listed within the BOM.

2.1.2.3 Instructions

Nothing is required to be built for the Raspberry Pi however, in order to set it up you must connect the RPi to an HDMI cord which is connected to a monitor. Then plug in the power cable to the RPi. The RPi should power up and you should be able to see this on the monitor. Thirdly plug in your USB keyboard and mouse to the RPi in the USB slots. Now you'll be able to type

and move the cursor on the Raspberry Pi. Finally, to have your RPi be connected to your computer in order to send messages and information between your dashboard software and RPi, connect your RPi to a Wi-Fi connection. The most direct way to do this will be to set up a mobile hotspot from your computer that the RPi can connect to and this will ensure the most direct connection between your computer and RPi without interruptions.

2.2 Mechanical

2.2.1 Relay

2.2.1.1 BOM (Bill of Materials)

- ¼” wire
- Inkbird SSR 25
- Male/ Female plugs
- solder
- copper wires with male and female pins (RPi connection)

2.2.1.2 Equipment list

- soldering iron
- wire cutters
- box cutter
- screwdriver
- pliers

2.2.1.3 Instructions

To wire the relay we used a PCB. The coil of the relay connected to the transistor. The transistor has 3 pins: the emitter, the base, and the collector. The coil is connected to the emitter. The base then connects to a resistor that connects to the RPi. The Collector connects to another GPIO pin on the RPi. To connect these pins, we used copper wire on the bottom side of the PCB and soldered them to their respective pins. To connect the main wire from the outlet to the relay we had to cut the positive wire in half and strip the ends. We then split the wires into two wires in a V shape. Next, we tinned the wires, so they wont fray while we soldered them to the relay. Finally, we solder the wires to four pins on the end of the relay opposite of the coil. Reference figures 3 and 4 from the list of figures for reference.

2.2.2 Solid State Relay

2.2.2.1 BOM (Bill of Materials)

- Solder
- InkBird SSR 25
- Wires
- Male/ Female Plugs

2.2.2.2 Equipment List

- Wire Cutters
- Box Cutter
- Soldering Iron
- Pliers
- Screwdriver

2.2.2.3 Instructions

Setting up the solid-state relay was much easier than our first attempt. To do this all you have to do is cut the positive wire, between the male and female plugs, in half and strip the ends. Then you split each end into two halves, making an V shape. Then you use pliers to pinch the ends together, making sure that a hole is left in the middle, to make an O shape out of the wires. Next you use the solder to tin the wires, so they maintain this shape. After the wires are prepped, all that has to be done is to attach them to the side of the solid-state relay that can hold a higher voltage, which is labeled clearly on the relay, and screw the wires down tight. Make sure the screws are as tight as possible to reduce the risk of the wires coming out. If the screws won't catch the threads, use pliers to flatten the wire. The screw is very short, so the thick wire makes it hard for it to reach the hole. Reference figure 5 from the list of figures for visual aid.

3 How to Use the Prototype

The function of this prototype is to control access to machines based on whether or not the user has the appropriate training. If the user doesn't have the training the electricity to the machine won't be able to flow and a message will be displayed on dashboard saying, 'Electricity Off'. If the user does have the training electricity will be allowed to flow to the machine and a message will be displayed on Dashboard saying, 'Electricity On'. The way these functions has the user inputting a student number on the dashboard UI on a tablet or personal laptop. This student number is then sent to the raspberry pi, which is connected across Wi-Fi to the tablet or personal laptop, which determines whether or not the user has training by comparing the student number given to a list of student numbers that do have the required training. Once the RPI determines whether or not the user has training it sends a message from a GPIO pin which is connected to the relay wiring telling it to either turn on or keep the electricity turned off. Simultaneously the RPi sends a message to dashboard as previously mentioned either saying 'Electricity On' or 'Electricity Off'.

To safely operate the prototype, make sure the Raspberry Pi is in a well-ventilated area as it has a tendency to overheat. Also attempt to use all same-size wires for the wiring to avoid a fire hazard when connecting the plugs to the PCB board which contains the relay. Another solution to avoid this fire hazard is to use a solid-state relay, however we were unable to get the prototype to work using this component. A third safety factor is when changing the wiring to the RPi or the relay while they're connected is to make sure the electricity is turned off and you can do this by inputting an invalid student number.

To install the prototype, first make sure that the Raspberry Pi is properly wired to the relay and that the RPi is connected over Wi-Fi to the tablet or personal laptop being used. Once that is

complete, plug the male plug into a power source on one end of the relay wiring and the other side must have the machine you're attempting to restrict access to plugged into the female plug.

4 How to Maintain the Prototype

To verify that the prototype was functional we had to test that current could flow through it, that the wiring was strong enough to withstand mechanical stress, and that the RPi and dashboard were able to communicate properly.

To test the flow of the RPi, a multimeter can be placed on the 5V pin and the ground pin. If electricity is flowing the multimeter should read 5V. When we performed this test on our RPi, at first it came up with a reading of 0, which lead us to believe we had shorted it out. Later on, we tested it again and it worked properly and gave the 5V reading.

To test the mechanical strength of the wiring we did a simple tug test. The main part of the device we had to worry about was the wiring around the relay because the wires we had were very thick which made it hard to connect to the relay. When we first soldered our board together, the thick wires were attached to the relay by soldering it to a smaller copper wire which then connected to the appropriate pin on the relay. This was unable to pass the tug test as either the thick wire would snap off or the smaller copper wires. because of this challenge we changed our board to a solid-state relay which only required the wires to be screwed into place rather than soldered. This made the wiring a lot stronger and allowed it to pass the tug test. To prevent the wires from coming out, the device should be stored in a box, like we had, and given slack, so the wires aren’t constantly under stress.

5 Conclusions et Recommendations for Future Work

Many lessons have been learnt along the design process of our final prototype which would further improve many different aspects of both our design and our efficiency. During our empathize and define phase, we learnt that constant and frequent communication with the client is very important, because the more we talk to our client, the more we are able to visualize their problems through their perspective. It is by empathizing with our client's needs that we are able to properly define design criteria and a problem statement. It is only at our client meeting where we discovered that our initial prototype would not be a great solution for our client, as they have already thought of a very similar idea. As a result, we were forced to do some divergent ideation to think of other ways to solve CEED's problems. More communication prior to this client meeting would have closer reflected the client's needs into our initial prototype. In other words, it is best to keep consistent communication with the client at hand.

Our ideate phase was one of our most valuable phases in terms of learning and improvement. After our client meeting, knowing that we had to find new ideas to solve CEED's problems, we learned about the importance of divergent ideation. Since our first prototype was not successful, we could now backpedal to the ideas which we thought of by using divergent ideation, which saved us time. By changing our design to be focused on the usage of machines in Brunsfield after our client meeting, we still encountered technical problems after our third prototype, causing us to ideate for more solutions. It is then where a CEED employee saw our issue and had proposed a great solution to solve our technical issues. Not only is asking people important, it is also important to ask the right people. The prototype phase, however, taught us how much each prototype differed from another, especially after we focused our design around Brunsfield. This

showed us how important flexibility was in terms of our design. As the problem changed, the prototype changed as well to address the client's problem.

While testing, it was evident that the earlier we started testing the more time we would have to adapt in case of failure, as our prototype had technical difficulties which we discovered later than planned. It is also important to consult the appropriate people before testing, as in our case, due to our poorly optimized electrical circuit, could have potentially caused a fire hazard.

APPENDICES

APPENDIX I: Design Files

Attached is the Dashboard code and the code for the Raspberry Pi.

https://makerepo.com/jward090/gng1103-relay-4-bytes-a8