Installation and User Guide for Venmo Vending Shield

Project by Paden Davis

Table of Contents

1. Purpose Statement

This document is written to provide a walkthrough and tutorial for interested parties to follow in order to set up and utilize the GitHub archive that has been provided. This document will include a step-by-step walkthrough of the creation/installation of physical hardware, as well as the setup of the server-side tools necessary to parse through emails to find the Venmo Purchase amount. This document does not include a walkthrough of bug fixing or error handling with these devices, as anything we did not encounter could not be anticipated and included. For any issues please submit an issue ticket within the GitHub.

2. Building and Design of the Shield

Forward

In the strictest sense, this shield is not necessary. The Arduino should be fully capable of plugging directly into the vending machine lines as they both run on five volts. I wouldn't however recommend taking this approach. The reason behind manipulating the signals and the wires to run through optoisolators is to separate the vending machine signal lines from the Arduino signal lines and to act as a buffer in case an issue occurs so that one or both of the devices do not break. This shield is not necessary to obtain the function of the device, and if required this section may then be skipped entirely, but it puts both devices at some risk and makes the longevity of the design weaker as a whole. The components necessary to make this modification are on the following page, and the device can take around an hour and a half to assemble on a protoboard. The design of the circuit can be seen below in Figure 1, with an example of my own implementation shown within Figure 2. Note that Figure 2 uses one optoisolator and one transistor as I was unable to acquire a second isolator, this is also not recommended but is better than wiring it without anything at all.

2.1 Figures

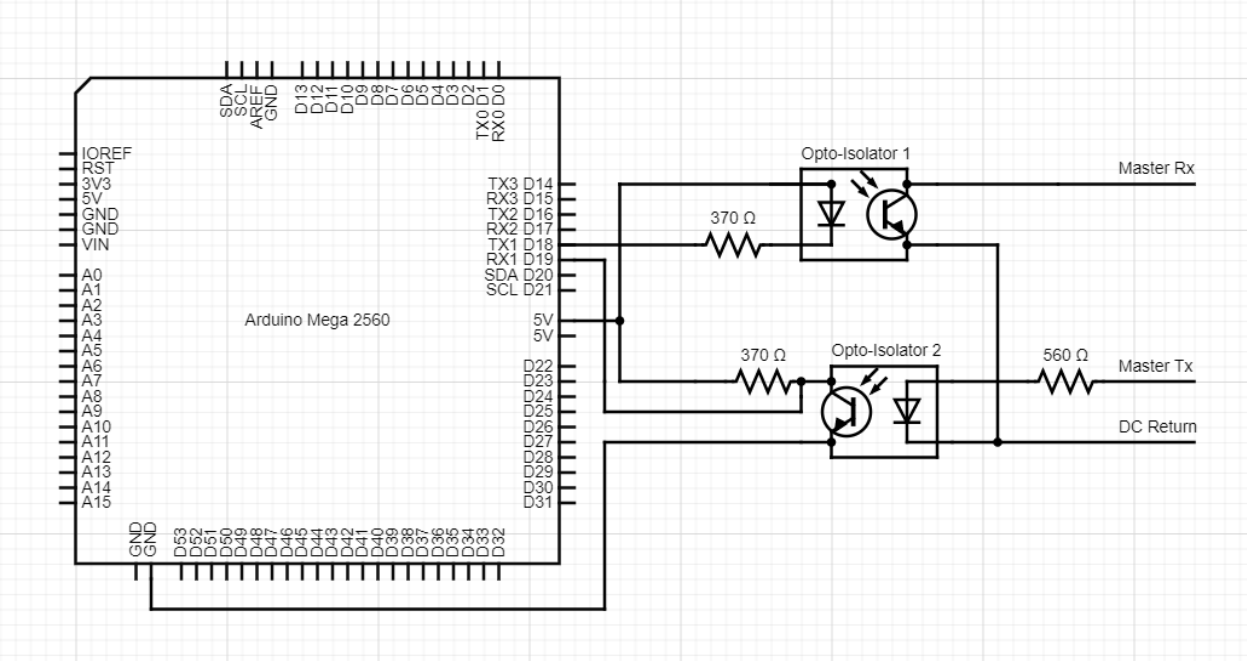

Figure 1: Circuit schematic for the Arduino shield and attachment to the Arduino pins

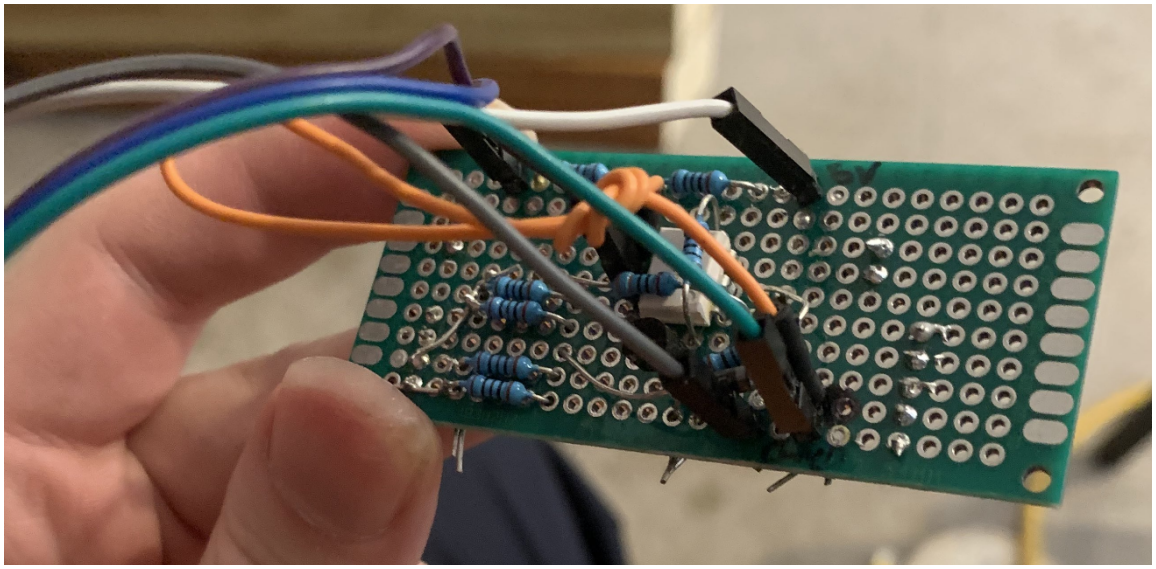

Figure 2: Picture of example breakout board containing the circuit. Note, multiple resistors were used to obtain the correct resistances for some connections, and one optoisolator was replaced with a transistor and diode, as we were unable to obtain a second. This is not recommended.

2.2 Parts List:

- One Arduino Mega
- One Arduino Ethernet Shield
- One appropriately sized protoboard
- Two 370 Ohm resistors
- One 560 Ohm resistor
- Two Opto-Isolators
- Six connecting wires
- Fourteen pin pegs

2.3 Construction of the Shield

1. Place the protoboard overlaying the Arduino Mega with the Ethernet Shield on, and mark out the holes to be used to attach the through-hole pin pegs.
2. Solder on the two 370 Ohm resistors attached to the Tx and Rx pins (pins 18 and 19 respectfully, see the wiring diagram in Figure 1).
3. Connect the Tx line with the resistor to pin 2 of the first optoisolator, connecting pin 1 to 5V either through the Arduino or by attaching another wire to connect to the 5V pin on the ethernet shield.
4. Connect the second resistor from the Rx pin to 5V as well, then connect the second optoisolator's pin 4 in parallel to the Rx pin.
5. Connect the second optoisolator pin 3 to ground
6. Bridge the first optoisolator's 3rd pin and the second optoisolator's 2nd pin.
7. Connect the first optoisolator's 4th pin to the master Rx line from the vending machine (it is helpful to do this, as well as the other two vending direct connections using removable connectors)
8. Connect the second optoisolator's first pin to the 560 Ohm resistor, and this to the vending machine's master Tx line.
9. Connect the second optoisolator's second pin to the vending machine ground, and connect this ground to the circuit ground.

2.4 Design Choices for the Shield

The design of the shield is made to be easily replicated using a 12x24 hole protoboard or protoboard of a similar size. Using this size, the shield is able to directly attach to the Arduino Mega's pins to access the Tx and Rx signal pins as shown in Figure 2. This size also allows for the use of the Arduino ethernet shield or a similarly sized wifi shield, allowing the Arduino to independently ping the server to read email messages. Because of the Arduino Mega's pin layout, however, the 5V and GND lines must also be run from the shield of choice using additional jumper wires (the white and grey lines in Figure 2).

The goal and reason behind the shield's necessity are to electronically isolate the two devices while allowing signal lines to pass between them. To this end, optoisolators were chosen as they provide a method to completely electronically isolate the two circuits with an extremely small delay between the two. One optoisolator handles the signals received, and one the signals sent. The ground line does not need isolation and is instead wired straight to the Arduino to establish a common reference for the devices

3. Installation of Shield

Forward

Now that the shield has been made, the next step is to interface the shield with the vending machine. In most vending machines, there will be a standard MDB connector running from the main board to the coin dispenser. It will be a six-pin connector with one pin missing, and will most likely follow this standard format in [5, Figure 3]:

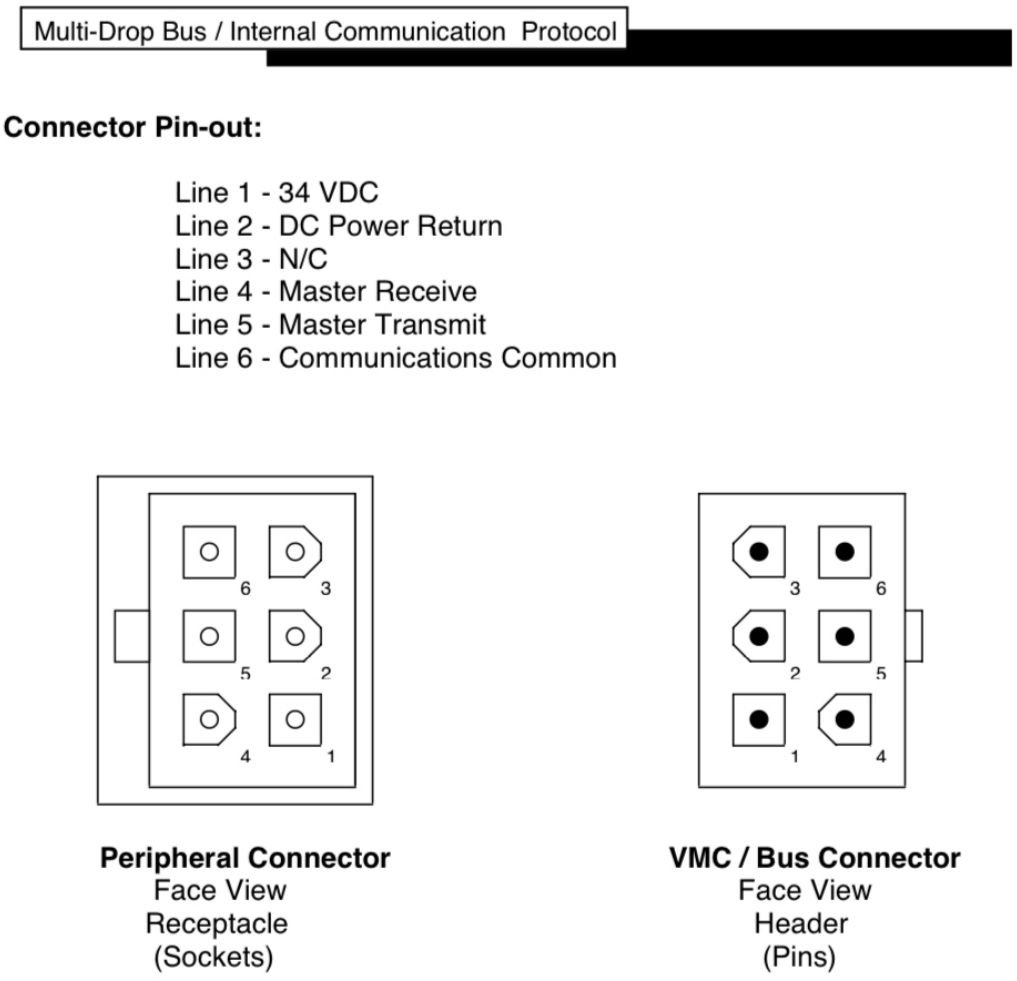

Figure 3: The standard wiring diagram for an MDB connector found in the MDB manual [5]

Splice three wires into the Master Receive, Master Transmit, and DC Power Return Pins and connect them as mentioned in the “Building and Design of the Shield” section. This should be all of the necessary modifications to the vending machine and should allow the device to operate through coin and bill as normal.

4. Proper Setup of Blynk and Arduino Code

Forward

Now you have the potential to interface with the vending machine, but do not currently have the ability to send or receive vending requests using the shield alone. In order to gain this ability, the code must be set up on the Arduino and a listener established using the Arduino Blynk library [4]. After this is set up, the user can also use the Blynk library to send fake requests to the Arduino and make sure that our code and wiring are working correctly. This is extremely useful to bugfix and ensures this section of the device is working correctly, as the server connection side tends to be somewhat difficult to get working.

4.1 Setup Instructions:

1. First, ensure that everything necessary to use blynk has been obtained. Use a phone to install the blynk app and make an account, then create a new project named whatever you like.
2. Install the blynk Arduino libraries found from Source [4]
3. Enter the new blynk project within the app and copy the auth code, then enter the Arduino program and replace the value of “auth_token” with the auth code that you are given. Make sure to save before compiling in case the IDE crashes.
4. Compile/Export the code from the Arduino IDE to the Mega board.

Now that the code should be compiled and uploaded to the Arduino Mega, plug the board into a barrel jack power supply and plug in the ethernet shield on top. Once powered and plugged into an ethernet connection, the blynk app should register that the device is online, in which case a string containing the format “time > name sent you $number” on virtual pin V9. The Arduino should then acknowledge the request by saying “http request received” followed by the request.

5. Proper Setup of Server-Side Code

Forward

Now that we have the ability to interface with the Arduino, we must set up the server-side code to be able to parse the Venmo email responses on the server. To do this we will be using the service (hosting service) in order to parse our emails and send them back to our script hosted within our server. This will then be sent over the blynk code we established earlier to allow the Arduino to properly identify the amount of money in each vend request.

5.1 Setup Instructions:

1. Create an account for this service and paste in the code given within the GitHub.
2. Once that has been completed, log into your server or dedicated device and upload the script given in the GitHub called “serverside.py”
3. Change the file “pw.txt” to hold the information corresponding to your email
    a. Server - Email server address
    b. Port - Email Server Port
    c. Username - Email Username
    d. Password - Email Password
    e. Auth_Token - Blynk Auth Token
    f. emailVerification - A string proceeding to help locate emailAuthentication
    g. emailAuthentication - A string found in emails from venmo to identify that an email is from the correct source
4. Use the following command (or follow a common guide such as this one labeled as Source [2] in references) to allow the script to run upon the device startup to ensure it is always running even after updates or power outages.
    a. `sudo update-rc.d serverside.py defaults`
5. Run the script using the following while in the directory of the script:
    a. `python3 ./serverside.py`

6. Description and Explanation of Code Sections for Modification

Forward

This section is made for those that wish to adapt or edit the code within this project to fit their own projects. The MDB code files that are included with the GitHub were created by Bouni and Alex Antonov from MateDealer [3] and were adapted for use within Vendmo.ino. As those files were not my creation, for more information on them please refer to the MateDealer GitHub in references. There are two files that will be addressed in this section: serverside.py and Vendmo.ino. Each file's approaches and strategies will be touched on within this section to help guide those who may wish to edit them in some way.

6.1 Description of File Uses

Serverside.py - This script is used to retrieve emails and parse them for information to send to the blynk host, and thereby to the arduino. The first thing this file does is attempt to establish a connection to the user's email server using the credentials provided within the pw.txt file. It then moves into a loop with a try-catch block to catch errors that arise from a failed connection to the email host. Within the Try block, the code selects the inbox to parse, and then searches for any messages containing the string "paid you". If an email with that string in its subject is located, the script pushes that to the next section of code which checks to make sure the email is unseen, and then parses the email for necessary information such as the date, amount paid, and authentication line. Once it obtains that information, the authentication line is compared to the emailAuthentification line from pw.txt to ensure someone is not trying to trick the system, and if they match, it proceeds to send the necessary data to the blynk service using the auth_token from pw.txt.

Vendmo.ino - This code is used to ping Blynk, receive commands, and then translate those commands into MDB signals. This file will have each function described individually:

setup() - This function initializes our Tx and Rx pins using setup_usart, then establishes a blynk connection and finds the current state of the vending machine. This should run once before the main loop.

loop() - This function first calls Blynk.run() which polls blynk to receive an update on if any money has been received. It then runs the mdb_cmd_handler() to wait and receive an MDB check from the machine, which waits until it receives an MDB command, translates it, and calls the appropriate MDB function to respond. Once the response is made, we then call the uplink_cmd_handler() function to return a response to the vending machine. The

dtoa() - A function used to convert a double to a character array. This is necessary to mesh the amount received from blynk (a double) with the vend command through the MDB library (takes a character array pointer).

vend() - Receives the amount to deposit into the machine and converts it using dtoa to hand to the cmd_approve_vend() function to add the coins to the vending machine's virtual wallet.

BLYNK_WRITE() - These methods take the various communications from blynk and turn them into arduino actions whenever a blynk.run() command occurs and blynk sends data. Each V# combination represents a signal on blynk's virtual pin corresponding to the number. V1 simply returns an update for the blynk side to display a change in the slider and to update the manual vend amount. V2 reproduces a vend() command based on the slider within the app when the button is pressed. V9 is an actual vend request from blynk, at which time the message is parsed for the necessary vending information, which the code then sends back as a confirmation message by using a BLYNK_WRITE(V5). The command then tells the machine to call the vend() function with the parsed amount (capped at $10 USD).

7. References

[1] “Arduino Mega 2560 Rev3 | Arduino Official Store.” https://store.arduino.cc/usa/mega-2560-r3 (accessed Apr. 25, 2021).

[2] “boot - Run bash script on startup,” *Raspberry Pi Stack Exchange*. https://raspberrypi.stackexchange.com/questions/15475/run-bash-script-on-startup (accessed Apr. 25, 2021).

[3] Bouni, A. Antonov, GitHub. 2020. MateDealer. Available: https://github.com/Bouni/MateDealer

[4] “Get started with Blynk in 5 minutes.” https://blynk.io/en/getting-started (accessed Apr. 25, 2021).

[5] *Multi-Drop Bus / Internal Communication Protocol,* NAMA, 20 N. Wacker Drive, Suite 3500 Chicago, Illinois. https://www.ccv.eu/wp-content/uploads/2018/05/mdb_interface_specification.pdf (accessed Apr. 25, 2021).