Tombola Controller

Hardware and Software Functional Requirements

By **Jesse Zondervan and Gary Twinn**

# Contents

[Tombola Controller 1](#_Toc151219951)

[Contents 2](#_Toc151219952)

[List of Figures 3](#_Toc151219953)

[List of Tables 4](#_Toc151219954)

[Overview 5](#_Toc151219955)

[Raspberry Pi 6](#_Toc151219956)

[Software Application 8](#_Toc151219957)

[Raspberry Pi System Installation and Configuration 12](#_Toc151219958)

[Operating system 12](#_Toc151219959)

[Software installation 12](#_Toc151219960)

[Run Operating System Updates 12](#_Toc151219961)

[Install PIP3 for Python 3.x installation 12](#_Toc151219962)

[Install Flask Libraries 12](#_Toc151219963)

[Install Minimal Modbus Package 12](#_Toc151219964)

[Download Tombola Controller Application 12](#_Toc151219965)

[Copy the python files to the Pi 13](#_Toc151219966)

[Nginx installation 13](#_Toc151219967)

[Gunicorn for Python 3.x installation 13](#_Toc151219968)

## List of Figures

[Figure 1: Connection Diagram 5](#_Toc151219969)

[Figure 2: Schematic of RPM Interface. 7](#_Toc151219970)

[Figure 3: Web status page 10](#_Toc151219971)

[Figure 4: API messages that are accepted 11](#_Toc151219972)

## List of Tables

[Table 1: MODBUS Registers 8](#_Toc151219973)

[Table 2: Tombola Controller Settings stored in the Json file. 9](#_Toc151219974)

# Overview

The UCL Tombola riverbed erosion simulator consists of a large (1m) sealed drum that can be filled with water and ball bearings to simulate gravel on the riverbed. Two blocks of concrete are used as a proxy for the underlying bedrock. As the drum rotates the water and ball bearings flow across the concrete blocks and erode them. The drum is powered by a 3 phase electric motor. The drum rotates about a hub taken from the rear of a Vauxhall Corsa automobile and contains an anti-lock brake sensor (ABS). The ABS sensor contains 48 magnets located evenly around the inside of the hub and a Hall effect transducer that signals as the magnets pass.

The Tombola Controller consists of a Raspberry Pi 3B single board computer ([Raspberry PI Foundation, 2020](#_ENREF_8)) and is linked to a Siemens V20 single phase to three phase invertor ([Siemens, 2023](#_ENREF_10)) which provides controlled power to the motor (Figure 1). The Raspberry Pi Communicates with the Siemens V20 via an RS-485 ([TIA, 1998](#_ENREF_14)) controller. Feedback from the Tombola drum is via an RPM interface that takes the signal from the Hall Effect sensor and passes it to a general-purpose input / output (GPIO) line on the Raspberry Pi to provide details on the rotational speed.

A diagram of a machine

Description automatically generated

Figure : Connection Diagram

# Raspberry Pi

Using a low cost single-board computers that support General Purpose Input Output (GPIO) and open-source programming languages is a cost effective approach to control laboratory equipment ([Twinn, 2023](#_ENREF_15)). The computer chosen was a Raspberry Pi model 3B, it is a single board computer that has a 4 core 1Ghz ARM 64bit CPU and 4Gb of RAM. A 32Gb MMC memory card provides persistent storage and serves as the main storage drive. The Raspberry Pi was configured to use the latest headless Debian 64bit operating system ([Software in the Public Interest, 2020](#_ENREF_11)). The minimal Debian OS distribution only includes the basic operating system and networking components require to run a as server, it has no graphical or desktop components that would consume processor and memory capacity. The Raspberry Pi has Python 3.11 installed ([Python Software Foundation, 2023](#_ENREF_7)) as the application to control the Tombola was written in Python.

The Raspberry Pi connects to the Siemens V20 via a RS-485 interface plugged into a USB port and access as a serval device at the address /dev/ttyUSB0. RS-485 is a two wore balances pair serial interface that supports a number of data rates ([TIA, 1998](#_ENREF_14)). The Siemens V20 has a default baud rate of 9600 which is adequate for this application and provides resilience to electrical interference. The Siemens V20 used the MODBUS command protocol ([Modbus Organisation, 2012](#_ENREF_4)) which consists of specifying and setting registers to set values for the motor such as rotational speed or direction and reading registers such as frequency and voltage.

Rotational feedback from the Tombola hub is provided by a hall effect sensor built into the hub (originally the ABS sensor) which connects to the Raspberry Pi GPIO via a custom built interface (Figure 2). The RPM is calculated by measuring the time for the Hall Effect sensor to detect 48 pulses. The Hall Effect sensor takes uses a 12 V power supply and provides an output voltage of 5.7 V when the sensor is not near a magnet and 8.0 V when the sensor is activated by a magnet. The interface takes the signal from the Hall Effect sensor and passes it to an NPN bipolar transistor with a high forward gain (Figure 2 TX1), this causes the transistor to switch on if the signal is above 6.5 V, and off if it below. As the output of the transistor is either 12 V or 0 V and the GPIO is expecting a signal of 3.3 V or 0 V an opto‑isolator (Figure 2 U1) is used, this consists of an integrated package that contains an LED and phototransistor, when current flows through pins 1 and 2, the LED will light and the transistor will switch on. Using an opto-isolator provides protection for the Raspberry Pi by isolating it from the 12 V circuit, even in the event of a component failure. The Raspberry Pi has GPIO channel 27 configured as an input with a rising edge detection, this generates an interrupt to the ARM processor as the input voltage on the GPIO moves from 0 V to 3.3 V. The circuit board also has a 5 V connection for the Raspberry Pi touch screen and an additional LED that lights when GPIO 17 is set to output and on, this is used to signal the Raspberry Pi is ready to receive RPM data.

Safety information is also fed back via a webcam attached to a USB port.

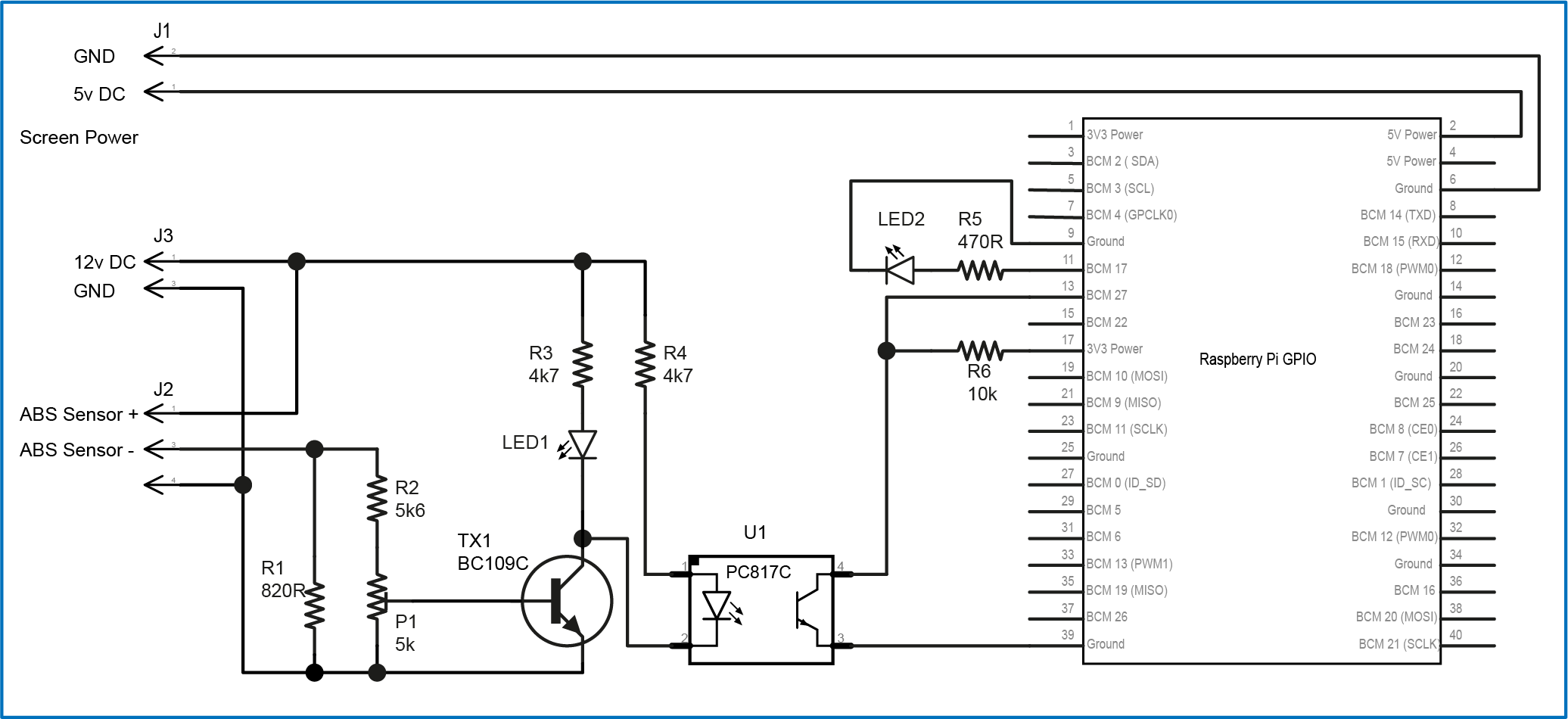


Figure : Schematic of RPM Interface.

5v power from the Raspberry Pi provides power to the LCD touch screen

# Software Application

The application is written in Python 3 and consists of four main modules:

1. A motor control module that connects to the Siemens V20 and command and query functions so send commands to the V20 and read data back
2. An RPM module that responds to the interrupt signals from the RPM Interface and calculates the Tombola RPM
3. A web application to provide a user interface
4. A camera module to stream live video from the web cam
5. A settings function to store the settings used by the modules above.

The application can be downloaded from <https://github.com/westerlymerlin/UCL-tombola>

The module that communicates with the V20 controller is the python class (MotorClass). The MotorClass utilises the MinimalModbus library for Python 3 ([Berg, 2023](#_ENREF_1)) to handle the MODBUS messages and uses a standard Linux address (/dev/ttyUSB0) for the RS485 adapter. Using MinimalModbus library, the motor class configures the port and then only needs to specify a start register and then send data to set the values or send a read command to read them back.

The MODBUS registers used to set the desired values for speed, direction and to enable the V20 are registers 40003 – 40006. The values returned from the V20 are in registers 40024 through 40034 (Table 1).

|  |  |  |
| --- | --- | --- |
| **Register** | **Description** | **Access** |
| 40003 | Frequency setpoint | Write |
| 40004 | Run enable | Write |
| 40005 | Forward/reverse command | Write |
| 40006 | Start command | Write |
| 40024 | Frequency output | Read |
| 40025 | Speed | Read |
| 40026 | Current filtered | Read |
| 40033 | Voltage output | Read |
| 40034 | Forward/reverse | Read |

Table : MODBUS Registers

The MotorClass has a function that can analyse commands that come in via the web app and then either send a command to the V20 or query registers.

The RPMClass contains an array of values for 3 x the number of magnets on the hub and each time there is an interrupt on GPIO 27, a function appends the time the interrupt was received to the array. Once there have been 3 revolutions of the tombola the time for the first reading is subtracted from the last reading to give the time for 3 revolutions. By using the time for 3 revolutions and averaging them, any small inconsistencies caused by moving weight (water and ball bearings). If the gap between two readings (1/48th of a turn) is over a set time (default is 2 seconds) it is assumed the tombola has stopped and an RPM value of zero is returned.

There is a settings function that allows specific settings to be changed, these are saved in the file “settings.json”. Using settings file allows the application to be customised for different applications or testing on different computers.

|  |  |  |
| --- | --- | --- |
| **Setting** | **Values** | **Description** |
| autoshutdown | true, false | Enable or disable the auto switch off at a set time |
| shutdowntime | hh:mm:ss | The time to shut down |
| port | com 1- 9  /dev/ttyUSB0 - 9 | The address of the RS485 controller |
| baud | 300 – 115200 | The RS485 port speed |
| stopbits | 0, 1, 2 | The number of stop bits in an RS-485 message |
| bytesize | 8 | The sise of a data byte on the RS-485 message |
| station | 1 - 1024 | The station address for the Computer on RS-485 |
| control\_offset | 40001 – 49999 | The start register for writing data to the V20 |
| reading\_offset | 40001 – 49999 | The start register for reading data from the V20 |
| read\_length | 1 – 1024 | The number of registers to read (from the read\_offset) |
| clear\_buffers\_after\_call | true, false | Clear the data buffer on teh RS-485 controller after reading data |
| clear\_buffers\_before\_call | true, false | Clear the data buffer on teh RS-485 controller before reading data |
| timeout | 0 – 60 | The number of seconts to wait after calling the V20 before abandining the message and rasining timeout error |
| rpm\_sensor\_GPIO | 1 – 28 | The GPIO chanel used to read the ABS sensor |
| rpm\_magnets | 1 - 1024 | The number of magnets to be read for a single revolution |
| rpm\_timeout\_seconds | 1 - 60 | The number of seconds between tow pulses to wait and assume the tombola has stopped |
| rpm\_active\_LED | 1 – 28 | The GPIO chanel to set to ON in order to light the Ready LED |

Table : Tombola Controller Settings stored in the Json file.

The main user interface is a web application designed around the Flask web application framework ([The Pallets Project, 2020](#_ENREF_13)). Flask is a Python framework that allows for rapid development and testing of Web applications without the programmer having to code the services that manage web connections. The flask application connects to a backend Python class to provide the individual functionality for the application. Although Flask framework contains a built-in web server, it is only suitable for use during application development as it is not robust enough for hosting a production service ([Relan, 2019](#_ENREF_9)). To provide a robust and secure service a Gunicorn webserver ([Chesneau, 2020](#_ENREF_2)) and Nginx reverse proxy ([Sysoev, 2020](#_ENREF_12)) are implemented. Gunicorn allows for a better security model than Flask, while Nginx handles the network connections and ensures only valid HTTP requests are routed to the Python application.

The main web application module (app.py) has the HTTP paths that are accessed to provide different pages, an endpoint for the status (read via JavaScript) and an Application Programming Interface (API) that accepts commands as Json messages ([Peng *et al.*, 2011](#_ENREF_5)) to allow for other applications to interact with it.

Access to the controller is via a web page which shows the status of the invertor and allows a user to set the rotational speed of the motor and stop the motor (Figure 3). The main web page uses JavaScript to update a number of values every second to give a near real-time view of the configuration and status of the V20 controller and tombola speed. The page shows the requested values that have been sent to the v20 controller as well as the received values back. The frequency is the value that sets the motor speed. The V20 expects the frequency to be set in units of 1/100 of a percent (0-10,000) but the query responds back with the frequency in Hz. The values for voltage, current that the V20 is supplying and an estimated motor rpm are also returned and displayed on the web interface. If the Tombola is to be left unattended an auto-stop function is available to switch off the motor at a set time.

User control is via four buttons:

1. The page has a “Start Tombola” button that requires the frequency to be entered. It will then send a forward(frequency) command to the V20.
2. The “Stop Tombola” button will stop the Tombola Motor.
3. The “Update Stop Time” button will allow you to change the stop time, the check box sets if the autostop is enabled.
4. The “Reset V20” button is used to reset the control word after a power failure.



Figure : Web status page

There are also a basic set of web pages generated by Flask to show the application log files, system log and webserver error logs for troubleshooting issues.

There is also an application programming interface (API) which gives additional access to the V20 as well as returning data from the RPM module. The API is not available from a normal web browser but can be accessed via standard API testing tools such as Postman or Insomnia ([Postman, 2020](#_ENREF_6); [Kong Inc, 2023](#_ENREF_3)). To use an API the users needs to connect to the /api endpoint and [POST] a Json message in the request.

|  |  |
| --- | --- |
| **Json message** | **Description** |
| {"setrpm": n.n} | Start the tombola running and hold it at n.n rpm (0.1 - 74.9 rpm) |
| {" setrpm ": 0} | Stop the motor |
| {"rpm": true} | Read the tombola RPM |
| {"rpm\_data": true} | Read the tombola timing data from 3 revolutions |
| {"write\_register": rr, "word": ww} | Write the word ww to the register rr |
| {"read\_register": rr} | Read the value from the register rr |
| {"stoptime": "HH:MM:SS", "autostop": true} | Set the controller to auto shut off at HH:MM:SS |
| {"stoptime": "HH:MM:SS", "autostop": false} | Disable auto stop |

Figure : API messages that are accepted

# Raspberry Pi System Installation and Configuration

## Operating system

Use BalenaEtcher to install the latest version of the bookworm-lite operating system onto a 32Gb MicroMMC card.

Connect via a USB keyboard and monitor and boot up the Raspberry Pi

Set the new username to tompi

Set a secure password

Run the sudo raspi-config command to:

enable ssh

disable Serial

disable 12C bus

change the GPU memory to 16Gb

set the hostname to byron-corelab

# Software installation

## Run Operating System Updates

Run sudo apt update

Run sudo apt upgrade

## Install PIP3 for Python 3.x installation

Run sudo apt install python3-pip

## Install Flask Libraries

Run sudo pip install flask --break-system-packages

## Install Minimal Modbus Package

Run sudo pip install minimalmodbus --break-system-packages

## Download Tombola Controller Application

Download from <https://github.com/westerlymerlin/UCL-tombola.git>

## Copy the python files to the Pi

Copy the files from GitHub to /home/tompi/

Copy the folder templates to /home/tompi/templates

Copy the folder static to /home/tompi/static

Create a folder /home/pi/logs

Copy the files /raspberry-pi/home/tompi to /home/tompi

Copy the file /raspberry-pi/home/tompi/etc/resolv.conf to /etc/resolv.conf

## Nginx installation

Run sudo apt install nginx

Copy the Github file

\raspberry-pi\home\pi\etc\nginx\nginx.conf

to

/etc/nginx/nginx.conf

Copy the GitHub file

\raspberry-pi\home\tompi\etc\nginx\sites-available\tombola

to

/etc/nginx/sites-available/tombola

Change directory to /etc/nginx/sites-enabled/

Run sudo rm default

Run sudo ln -s /etc/nginx/sites-available/tombola

## Gunicorn for Python 3.x installation

Run sudo apt install gunicorn3

Copy the GitHub file

\raspberry-pi\home\tompi\etc\systemd\system\gunicorn.service

to

/etc/systemd/system/gunicorn.service

Run sudo systemctl enable gunicorn

Run sudo systemctl start gunicorn

If flask is installed, the python files are in the /home/tompi directory, gunicorn3 is installed and configured and nginx is installed and configured the web service should be running and the site will be accessible on http://*ip address of the server*

References

Berg, J. (2023) *MinimalModbus library for Python*  The MinimalModbus Developers. Available online: <https://pypi.org/project/minimalmodbus/> [Accessed.

Chesneau, B. (2020) *Gunocorn, Python WSGI HTTP Server for UNIX*. Available online: <https://gunicorn.org/> [Accessed 2020].

Kong Inc (2023) *Kong Insomnia*  Kong Inc. Available online: <https://insomnia.rest/> [Accessed.

Modbus Organisation (2012) *MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3*, 2012. Available online: <https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf>.

Peng, D., Cao, L. & Xu, W. (2011) Using JSON for data exchanging in web service applications. *Journal of Computational Information Systems*, 7(16), 5883-5890.

Postman (2020) *Postman - an API platform for building and using APIs*  Postman, Inc. Available online: <https://www.postman.com/> [Accessed 2020].

Python Software Foundation (2023) *Python 3 Programming Language*. Online. Available online: <https://www.python.org> [Accessed March 2023].

Raspberry PI Foundation (2020) *Raspberry PI Model 4B Reference*. Available online: <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/> [Accessed July 2020].

Relan, K. (2019) Deploying Flask Applications, *Building REST APIs with Flask: Create Python Web Services with MySQL*. Berkeley, CA: Apress, 159-182.

Siemens (2023) *SINAMICS V20 - The compact basic performance converter*, 2023. Available online: <https://www.siemens.com/global/en/products/drives/sinamics/low-voltage-converters/standard-performance-frequency-converter/sinamics-v20.html>.

Software in the Public Interest (2020) *Debian Operating System*, 2020. Available online: <https://www.debian.org/>.

Sysoev, I. (2020) *NGNIX open source web server*. Available online: <https://www.nginx.com/> [Accessed 2020].

The Pallets Project (2020) *Flask is a lightweight WSGI web application framework*. Available online: <https://palletsprojects.com/> [Accessed 2020].

TIA (1998) *1998: Electrical characteristics of generators and receivers for use in balanced digital multipoint systems.*Telecommunications Industry Association.

Twinn, G. (2023) Combining low-cost single board computers with open-source software to control noble gas extraction lines. *MethodsX*, 10, 101974, DOI: <https://doi.org/10.1016/j.mex.2022.101974>.