**Cub Car Timer – The Details:**

Overview – The Cub Car timer is a project designed to run and manage cub car (pinewood derby) races to help run them in a fair and fun fashion.

Having been involved with CubCars over the past few years I have noticed that CubCar rallies tend to run one of two ways:

1. Kids are given a piece of paper and run around from track to track racing whomever they feel they can beat with their car. The count of wins and losses are totaled and the most number of “wins” move onto the next round. Some cars which “may” have made it into the later rounds may not have made it into those later rounds as they were eliminated by kids who figured out how to “game” the races and would only challenge cars they thought they could win against. The pro of this type of approach is that it allows for multiple tracks to be running concurrently, (but only 2 cars race at a time). If multiple tracks are used then kids quickly figure out which track their car runs ‘best’ on and try to race on that track as much as possible.

OR

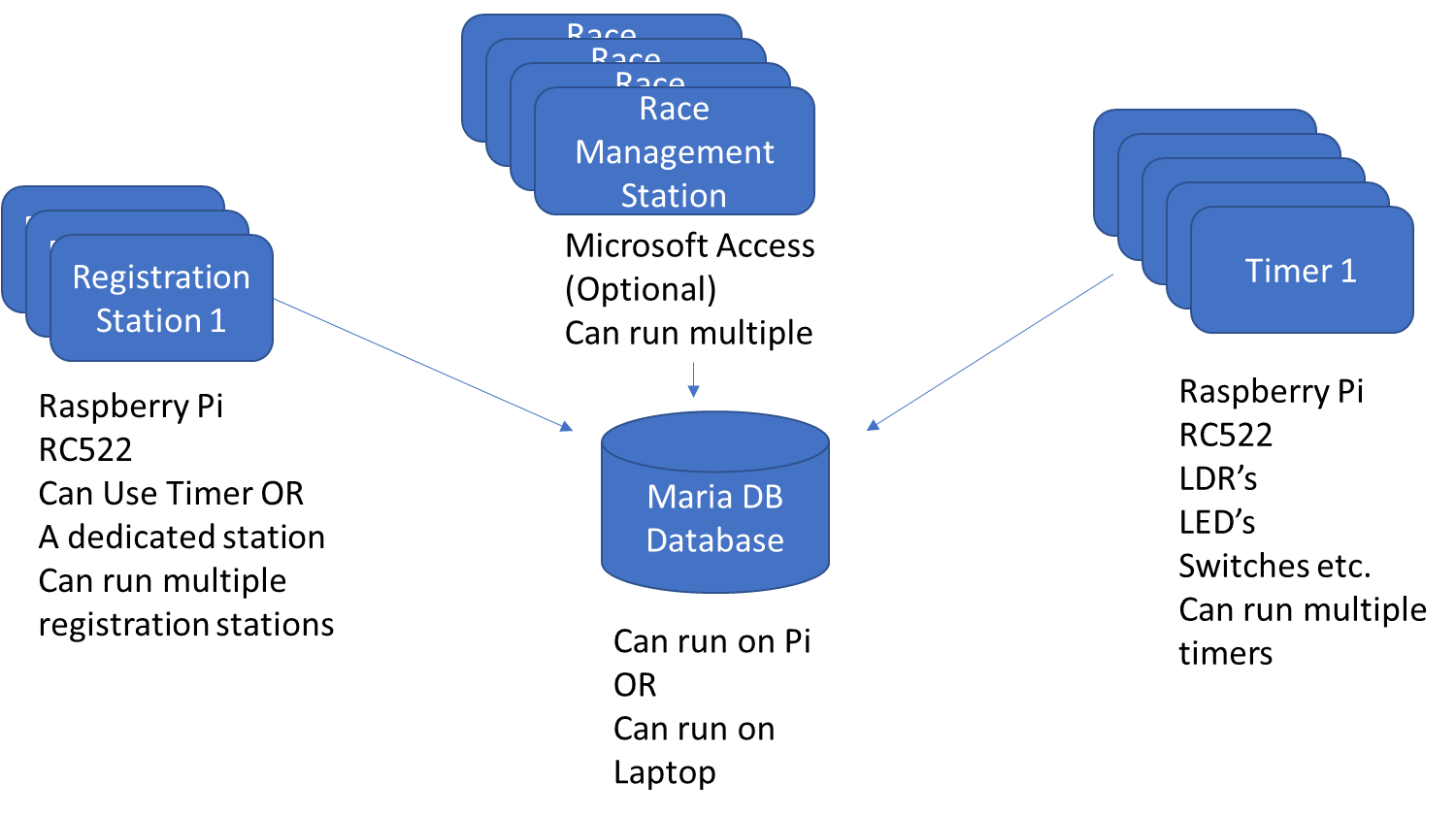
1. All the racers are plopped into spreadsheets which do pairings of racers which then determines who races who when. Often this means that only a single track is used to support and manage the races.

The approach of utilizing this timer with the RFID tags and the database behind the timer and a bit of programming allows for a very different approach to managing the races.

Although you don’t have to use this approach with the timer here is a much more fair approach. It is not about placing, it is about time.

1. Kids can run as many races as they would like against as many cars that they would like, so long as they run at least 5 (variable #) races on each track.
2. The cumulative time for the 5 fastest races on each track are totaled and ranked by shortest time.
   1. If multiple tracks are used then the cumulative time across each track is done and a combined ranking is done to determine the winner.

The following is the layout of how the various bits and pieces of the solution can be assembled together:



**Registration Station:** Raspberry Pi with an RFID pad on it which registers in the database the Car Name and the Name of the Racer and ties it the RFID tag on the car. (This can be run on the timer in between races) If you have a dedicated registration station you can use a pi zero as it is not a particularly demanding application. Note: If you are running a pi Zero you cant put MariaDB on it. (Not enough horsepower)

**Timer:** This is the main unit that runs all the races and is the heart of everything. This runs on a raspberry pi.

**Race Management Station**: This is an optional piece of the solution but it allows for a whole bunch of additional information to be collected and managed and supports running all the reporting for the solution. (As Maria DB itself does not have a GUI) It has a bunch of incomplete features in it. In future iterations of the solution this would get replaced with a web app. (Likely PHP, running on NGINX, which can also be hosted on a pi) today this must run on a laptop that has a Microsoft Access license on it. All of the tables in the MS-Access database are linked tables to the MariaDB database and do not store anything locally in MS-Access. Because the tables are linked you can have multiple race management stations running although likely at most all you will need is 1.

**MariaDB:** This is where all the information on the races and racer information is stored. Although there are a number of tables in the database the only two that matter for running and managing races are *racerinfo* and *raceresults.* The *raceresults* table is heavily denomalized so that is can be exported into Excel and have all the information in it to support analysis and further reporting.

The MariaDB can be hosted on the timer, can be on a dedicated raspberry pi or can run on a laptop. Although the solution can run without the database or the RFID pad. (If you use the RFID pad then you need the database)

The timer was developed using a Raspberry PI v3 with a standard Raspbian Stretch install on it. The project likely could run on a V2, however I don’t have one that I can validate / test on.

All the sensors are directly connected via the GPIO port. The program and controller is setup for 3 tracks, but could be expanded for 4 tracks. (Code would need help to expand to additional tracks)

The goal of the project is to build the entire solution for less than $100.00.

The solution can run headless (no screen or keyboard directly attached) or connected to a touchscreen or utilizing an external monitor.

The following is the circuit diagram and parts list.

|  |  |  |
| --- | --- | --- |
| 15 - High Power LED’s | 3 - 10K Light Dependent Resisters | 12 – 300 OHM Resisters |
| 1 – Momentary Push Button | 1 – Raspberry PI 3.0 | 1 - 40 PIN Header (Optional) |
| Strip Board or PCB | 3 - DB 9 Connector | 1 - 2 wire Extension Wire (Length of Track) (Speaker Wire) |
| 1 – RCA (track button Cable) | 1 – Raspberry PI Touch Screen (Optional) | 1 – MAX7219 LED Array (Optional) |
| 1 – RC522 RFID Reader (Optional) | 1 – 5 V PI compatible Relay board (Optional) | 1 - RS232 9 wire M-F cables (Length of Track) |
|  |  |  |

NOTE: During testing you can use Pushbutton Switches instead of LDR’s if you don’t have any LDR’s yet.

**Controller Assembly Notes**

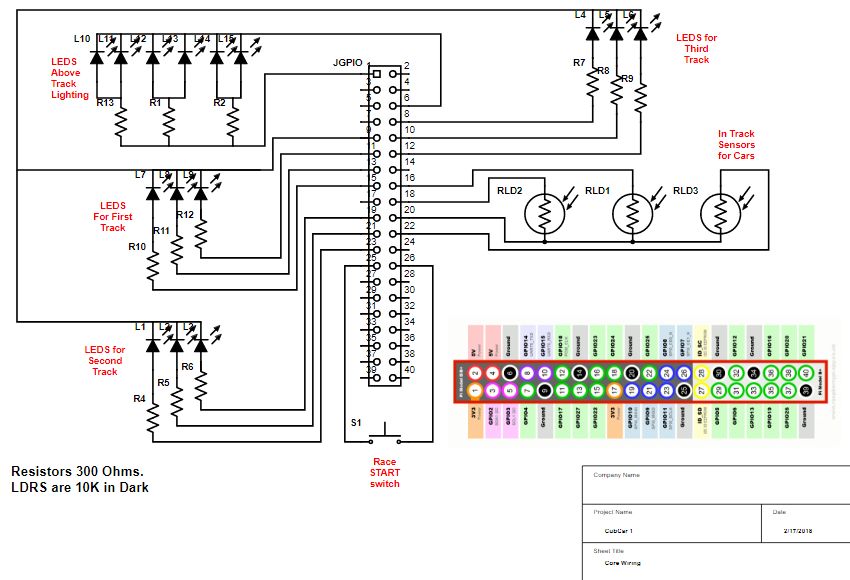
For the three different versions of the project we used different materials and approaches for building the controller box. For the first version a block of food grade plastic was cut up and a generous amount of epoxy was used to glue the pieces together and the electronics were hot glued into the box.

For the second generation of controllers we utilized a 3D printer to print out panels and bezels and riveted them to a metai project box to give a much more finished look.

Two big improvements in the 2nd generation of the controller is moving the LED bank to the top of the controller (so that they can be seen from all angles) and printing out a “bump out” on the back of the 2nd controller which houses the raspberry PI which gives access to all the port on the PI. Two factors you will want to consider for your enclosure.

**Electronics**

This circuit is the basic “Heart” of the project. The PINS that the LED’s and LDR’s are on are customizable depending on your needs. The wiring diagram does not show the MAX7219 or the RC522 RFID reader.



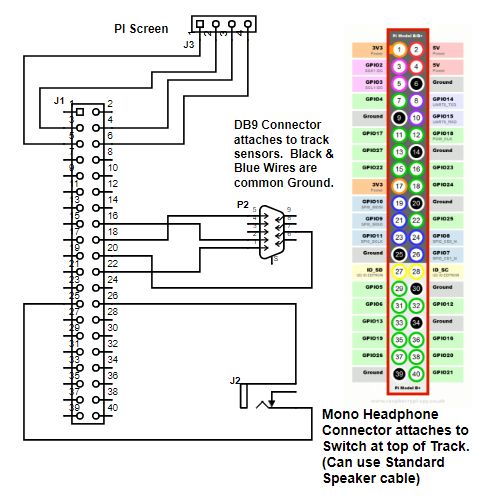
Note: The GPIO pins that the LEDS, LDR’s and switch are on are all configurable. Using the Legend you can use ANY of the PINS that are the same colour for the same purpose if you have conflicts with new devices you want to introduce.

Here is the latest wiring list for version 3 of the project.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Pi PINS** | **Usage** | **RFID PAD** | **RELAY BOARD (Future)** | **BUTTONS** | **TRACK LIGHTS** | **BANK LIGHTS** | **Track Switch** | **LDR** | **MAX 7219** | **Notes2** |
|  |  | **7** | **3** | **3** | **2** | **10** | **2** | **4** | **5** |  |
| 1 | 3.3 V |  |  |  | 3.3V |  |  |  |  |  |
| 2 | 5 V |  | 5V |  |  |  |  |  | 5V |  |
| 3 | GPIO02 |  |  |  |  | GPIO02 |  |  |  |  |
| 4 | 5V |  |  |  |  |  |  |  |  |  |
| 5 | GPIO03 |  |  |  |  | GPIO03 |  |  |  |  |
| 6 | Ground |  |  |  | GND |  |  |  | GND |  |
| 7 | GPIO04 |  |  |  |  | GPIO04 |  |  |  |  |
| 8 | GPIO14 |  |  |  |  | GPIO14 |  |  |  |  |
| 9 | GROUND |  |  |  |  |  |  |  |  |  |
| 10 | GPIO15 |  |  |  |  | GPIO15 |  |  |  |  |
| 11 | GPIO17 |  |  |  |  | GPIO17 |  |  |  |  |
| 12 | GPIO18 |  |  |  |  | GPIO18 |  |  |  |  |
| 13 | GPIO27 |  |  |  |  | GPIO27 |  |  |  |  |
| 14 | GROUND |  |  |  |  |  |  |  |  |  |
| 15 | GPIO22 |  |  |  |  | GPIO22 |  |  |  |  |
| 16 | GPIO23 |  |  |  |  |  |  | GPIO23 |  |  |
| 17 | 3.3V | 3.3V |  |  |  |  |  |  |  |  |
| 18 | GPIO24 |  |  |  |  |  |  | GPIO24 |  |  |
| 19 | GPIO10 | GPIO10 |  |  |  |  |  |  | GPIO10 | Investigate |
| 20 | GROUND | GND |  | GND |  |  |  | GND |  |  |
| 21 | GPIO09 | GPIO09 |  |  |  |  |  |  |  |  |
| 22 | GPIO25 | GPIO25 |  |  |  |  |  |  |  |  |
| 23 | GPIO11 | GPIO11 |  |  |  |  |  |  | GPIO11 | Investigate |
| 24 | GPIO08 | GPIO08 |  |  |  |  |  |  | GPIO08 | Investigate |
| 25 | GROUND |  |  |  |  |  | GND |  |  |  |
| 26 | GPIO07 |  |  |  |  |  | GPIO07 |  |  |  |
| 27 | ID\_SD |  |  |  |  |  |  |  |  |  |
| 28 | ID\_SC |  |  |  |  |  |  |  |  |  |
| 29 | GPIO05 |  |  | GPIO05 |  |  |  |  |  |  |
| 30 | GROUND |  |  |  |  |  |  |  |  |  |
| 31 | GPIO06 |  |  |  |  |  |  | GPIO06 |  |  |
| 32 | GPIO012 |  |  |  |  |  |  |  |  |  |
| 33 | GPIO13 |  |  | GPIO06 |  |  |  |  |  |  |
| 34 | GROUND |  |  |  |  | GND |  |  |  |  |
| 35 | GPIO19 |  |  |  |  |  |  |  |  |  |
| 36 | GPIO16 |  |  |  |  |  |  |  |  |  |
| 37 | GPIO26 |  |  |  |  |  |  |  |  |  |
| 38 | GPIO20 |  |  |  |  |  |  |  |  |  |
| 39 | GROUND |  | GND |  |  |  |  |  |  |  |
| 40 | GPIO21 |  | GPIO21 |  |  |  |  |  |  |  |

Note: you cannot have both the RFID PAD and the MAX LED matrix at the same time as they conflict with wiring and software. (Supposedly SPI supports two devices on the PI3, but I haven’t been able to make that work)

The following is the external wiring for the first generation of the solution. (Screen, Sensors, Remote start Button)

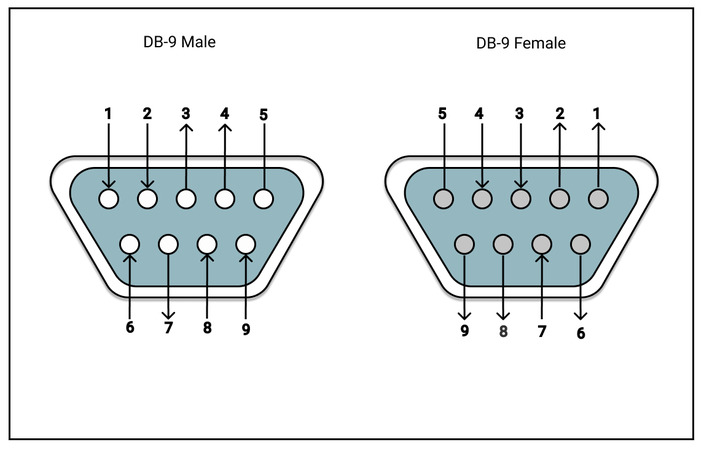
Not shown on the diagram are the wirings for the internal LED boards, LED Matrix, RFID PAD or Relay.

**Internal Wiring**

On the first version of the project I built small PCB’s for the LED’s and the Resistors, and put header pins on all the boards. These took far too long to build. For the second generation I built 1 PCB board. For the third generation I just soldered resistors directly to the LED legs and cut down the amount of time for assembly.



There are a LOT of wires to deal with and manage so you may want to build a colour legend to help keep things straight. For example here is how I wired all the DB9 connectors.



|  |  |
| --- | --- |
| **DB9** | **Diagram Colour** |
| 1 | Purple |
| 2 | Green |
| 3 | Orange |
| 4 | Brown |
| 5 | Black |
| 6 | White |
| 7 | Blue |
| 8 | Yellow |
| 9 | Red |

**MAX 7219**

Adding a Max7219 Matrix:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Board Pin | Name | Remarks | RPi Pin | RPi Function |
| 1 | VCC | +5V Power | 2 | 5V0 |
| 2 | GND | Ground | 6 | GND |
| 3 | DIN | Data In | 19 | GPIO 10 (MOSI) |
| 4 | CS | Chip Select | 24 | GPIO 8 (SPI CE0) |
| 5 | CLK | Clock | 23 | GPIO 11 (SPI CLK) |

Unfortunately the Max7219 matrix conflicts with both the integrated screen and the LED Bank, so for the 2nd generation I did not put in a touch screen but used an external monitor.

The Max7219 and the Pi Screen share a ground PIN (PIN 6)

Here is the code you need to run to install the MAX libraries for the LED Matrix.

sudo -H pip install --upgrade --ignore-installed pip setuptools

sudo usermod -a -G spi,gpio pi

sudo apt install build-essential python3-dev python3-pip libfreetype6-dev libjpeg-dev libopenjp2-7 libtiff5

sudo -H pip install --upgrade luma.led\_matrix

**LDR’s – (Light Detecting Resistors)**

I originally bought a 20 pack of assorted LDR’s from amazon and discovered that only a few of them actually work on the raspberry PI for this purpose. (Some were too sensitive, some needed way too much light to dark to actually trigger, some I could never get to trigger.) You can either buy a big assortment pack and trial and error your way through or buy one of the two types listed below.

Gl5549

Gl5537

(Note: Even if you buy one of these two types you will want to test them prior to installation as they have a lot of variability in their functionality)

**RF ID PAD**

To be able to install the RF-ID PAD you need to execute the following on your raspberry pi:

sudo apt-get update

sudo apt-get upgrade

sudo apt-get install python3-dev python3-pip

sudo pip3 install spidev

sudo pip3 install mfrc522

|  |  |  |  |
| --- | --- | --- | --- |
| **RS522 Header** | **Diagram Colour** | **PI Header** | **Notes** |
| 3.3 V | Red | 17 | 3.3V |
| RST | Brown | 22 | GPIO25 |
| GND | Black | 20 | Ground |
| IRQ |  |  | Not Connected |
| MISO | White | 21 | GPIO9 |
| MOSI | Grey | 19 | GPIO10 |
| SCK | Maroon | 23 | GPIO11 |
| SDA | Blue | 24 | GPI08 |

**Relay Board**

The relay board is a planned future enhancement whereby external light tree can get triggered to countdown the race. This runs on a completely different circuit. It is a 3 wire relay board.

|  |  |  |  |
| --- | --- | --- | --- |
| **Relay Board** | **Diagram Colour** | **PI Header** | **Notes** |
| 5V+ | White | 2 | 5V |
| GND- | Black | 9 | Ground |
| SIG | Grey | 16 | GPIO23 |

**MariaDB**

The third iteration of the program (under development) intents to use MariaDB to store racer information, car information and race results. The MariaDB can sit on either the raspberry PI or can run on a laptop. By using a database to store the information it allows for several timers to all run on separate tracks and to aggregate results in some really interesting ways.

You can find instructions on how to download and install MariaDB on the web so I wont cover that.

Note you will need to install the mySQL connector to make this work.

sudo pip3 install mysql-connector-python

A note about MariaDB and ROOT.

You cannot use root if you want to access the database remotely, you will need to create another user and grant it all the necessary privileges for that to function.

create user cubcaradmin identified by 'XXXX';

grant all privileges on \*.\* to 'cubcaradmin' ;

If you have MariaDB installed on a raspberry pi and you want to access it remotely you will also have to edit:

/etc/mysql/mariadb.conf.d/50-server.cnf

Change bind address to 0.0.0.0 instead of 127.0.0.1

**The Program**

I wrote the project originally in Python2, but upgraded to Python3 as part of the 3rd iteration.

Because I have been through several different versions and configurations of the timers all of the core settings for the timer are stored in an .ini file (Older versions called it settings.ini, after version 3 it is called RMSettings.ini).

Within this file all the core attributes of the project can be changed and adusted. The file is broken into several sections:

* Settings – This is information on how things are wired, debounce time for switches, Unit # etc.
* Dbase – These are all the settings for the database
* Races – This is information on fastest racer, what heat, etc.. (Mostly obsolete now that database is in place)

The GPIO’s identified in the settings file are the physical PIN numbers, not the GPIO names.

[Settings]

GPIO\_relay\_1 = 16 ## PIN for optional external relay

GPIO\_track\_switch = 29 ## PIN for switch at the top of the track

track\_switch\_bounce\_time = 600 ## Debounce time for Switch at top of track

GPIO\_car\_select\_switch = 31 ## PIN for optional switch in RFID PAD

car\_select\_bounce\_time = 600 ## debounce for car select switch (may need to tune)

GPIO\_pad\_switch = 33 ## PIN for secondary switch on RFID PAD (Future)

pad\_switch\_bounce\_time = 600 ## debounce for secondary switch on RFID PAD (Future)

max\_race\_time = 12 ## How long can a race be, will want to adjust

min\_race\_time = 1 ## How short can a race be. (DO NOT SET TO ZERO)

GPIO\_ldr\_lane\_1 = 16 ## PIN for Light Resistor Track #1

GPIO\_ldr\_lane\_2 = 22 ## PIN for Light Resistor Track #2

GPIO\_ldr\_lane\_3 = 18 ## PIN for Light Resistor Track #3

leds = 12,10,8,15,13,11,21,5,3 ## PINs for the LED’s on the light bank

unit = 1 ## What Unit # is this, Change this if multiple times

matrix\_yn = N ## Using Max 7219 Y/N?

[dbase]

host=192.168.100.251 ## What is the IP of the computer where database is?

user=cubcaradmin ## What is the name of the account to access DB

passwd=XXXXX ## Password for the database

database=attendancesystem ## Name of the database

[Races] ## Information that updates constantly

gfastestlane = 2 ## what lanes was fastest

gfastesttime = 2.8618221283 ## what was the fastest time

gfastestracecounter = 6 ## How many races run today

race\_counter = 0 ## What race # are we at

heat = 1 ## What heat

In future will be changing the program to prompt for database password on load (better security practice)

When starting a new set of races reset all the values in that section to 0 before starting the program.

**Starting the program:**

The program will update both to a terminal screen as well as to a GUI.

The screen and raspberry PI both have separate power supplies, the screen needs to be plugged in first then the PI.

At this point the program does not autostart so it needs to be manually started.

1. Open up Terminal Window
2. Go into cubCar directory - “***cd cubCar***”
3. List directory “***ls***”
4. Start the program “***sudo python3 RaceManager.py***”

**Exiting out of the program**:

Because the program waits for events to happen on the GPIO to exit properly out of the program you need to do the following:

1. In GUI press on the X in the top corner. (Wont look like you have done anything)
2. Swap to the terminal window.
3. Hit CTRL-X
4. Depress and release the remote trigger button

Program should end and cleanup the state of all of the LED’s etc. (Note: If you don’t exit out of the program properly you run the risk that the GPIO will be left in an inconsistent state and future programs will behave erratically.)

**Troubleshooting**:

Three main issues exist in the program today:

1. If room is too bright then ambient light will not allow for the light sensors to trigger. The track works best in a semi dark space. (Move away from windows etc.)
2. The GUI framework consumes 100% of CPU for the first few minutes it runs. You will want to wait a few minutes before running first races which “count”
3. GUI may become unresponsive. Unknown what the root cause is. Restart the program to correct.
4. Cars run down track, but race does not “start” – Most likely cause is that start button was triggered improperly. For future race need to wait longer between pressing the start button and releasing it. (See section on start button)

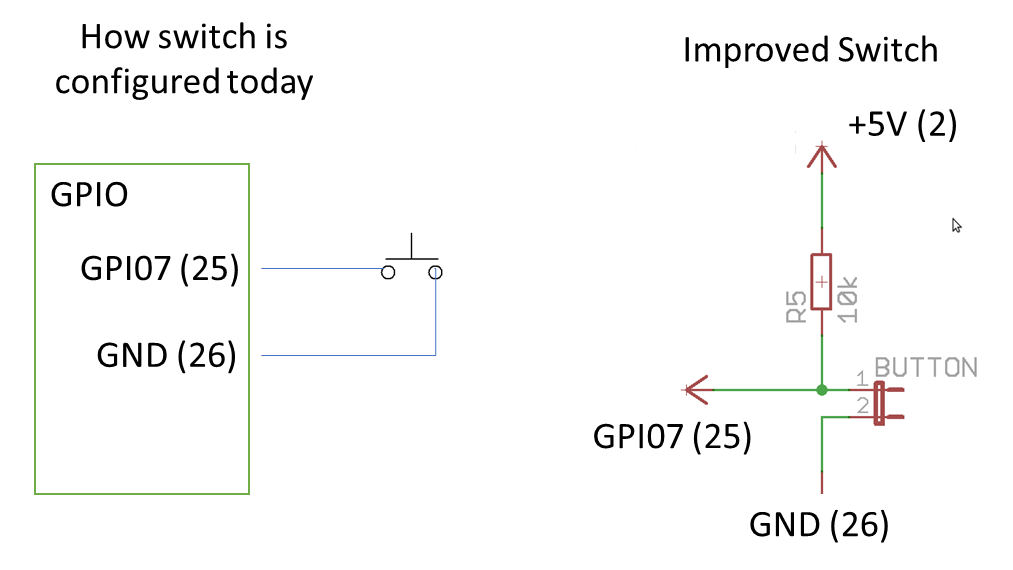
**Running a Race:**

* On the GUI the Red circle means that you can prep the track and load cars. Do not load cars if the GUI showing a green circle. (That means previous race has not yet finished)
  + Flip up the track start which will depress the race start button.
  + Put first car on RFID pad, and lights over 1st track bank will light up.
  + Hit button 1 on RFID pad
  + Put next car on RFID pad and light over next track will light up
  + Hit button 1 on RFID pad
  + Repeat for all the cars.
  + Wait a few seconds before releasing cars down the track.
  + Circle turns green indicating that race has started
  + Lights will quickly flash on controller when race has started successfully.
  + Finishing indicators will show which cars completed in 1st, 2nd and 3rd.
* If only 2 cars are racing or car(s) derail then race will automatically end after 20 seconds. (Circle turns Red indicating race is finished.)

**Understanding the Start Switch**.

The program will stop and wait for 2 events on the start button at least 1 second apart. (Button Push Event, followed by a button release event).

In the event of ongoing issues with the reset switch not functioning properly the following changes can be made.



Future Enhancements:

The programming itself is horrible as it is a cobbling together of a few different programs without a lot of time spent on optimization. (No Consistency with Naming Conventions, inefficient code etc.)

For future: Use of threading and callbacks to make GUI more responsive.

For future want to change the RFID pad to use I2C instead of SPI as it allows for longer cables to the RFID PAD. (50’ is really pushing it with RFID pad)

May want to look at LED multiplexing to allow for much more interesting visuals.

GUI is horrible and needs complete redesign.

`

