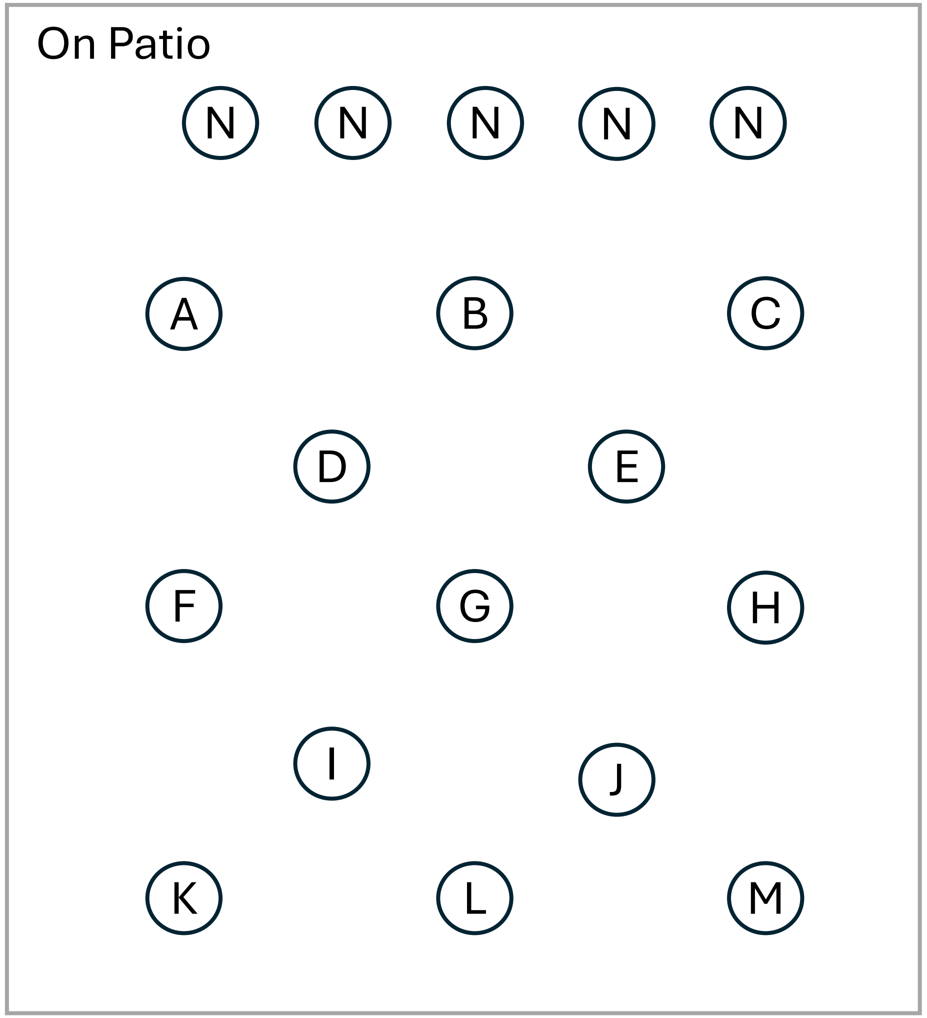
Splash Pad Controller Notes

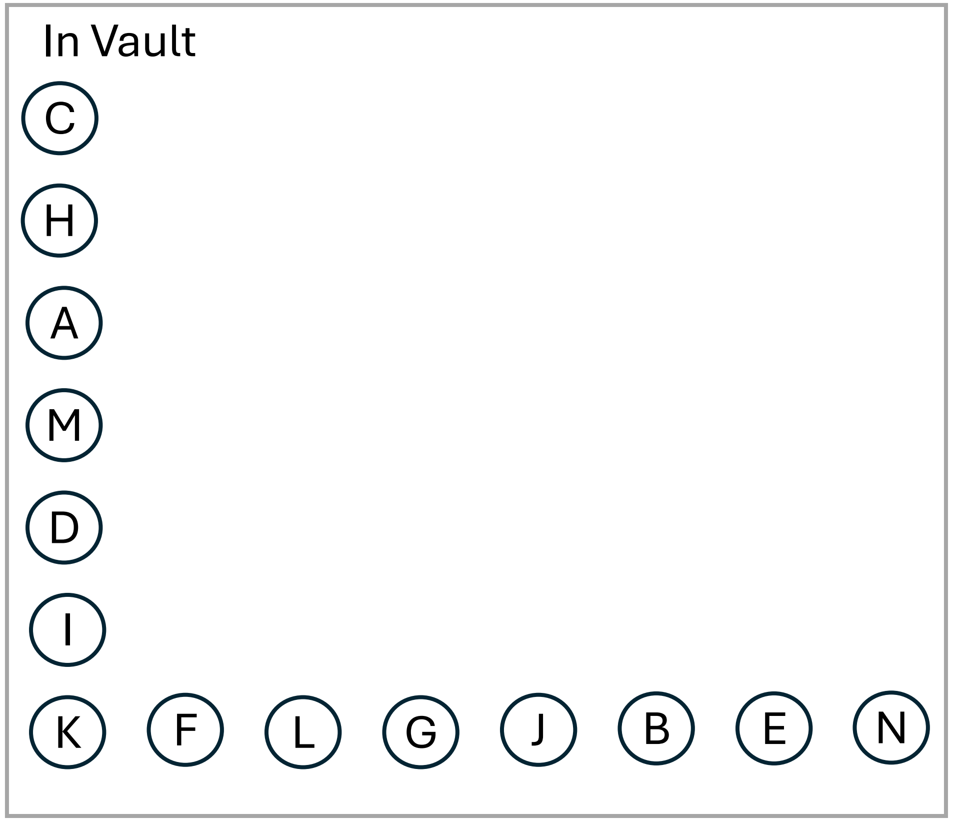
Aug 13, 2024

Rebuilding The Vault Machinery

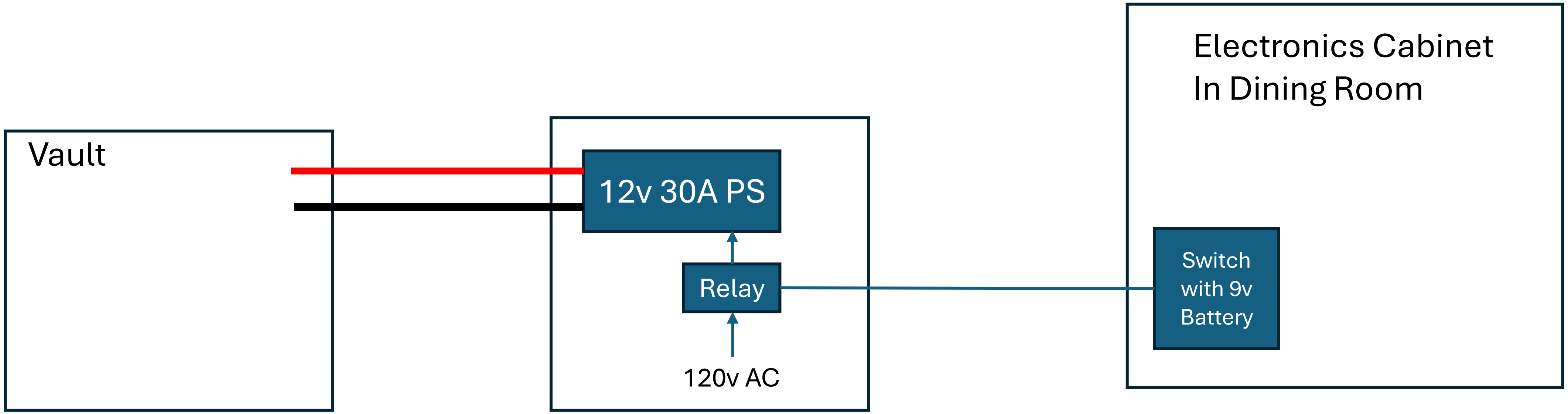
- Old System based on Raspberry Pi 3 and two relay banks of 8 relays each.
- Decided to completely redo electronics, but keep the design based on relays instead of a custom circuit board with Mosfets.

Splash Pad Layout





Power Control

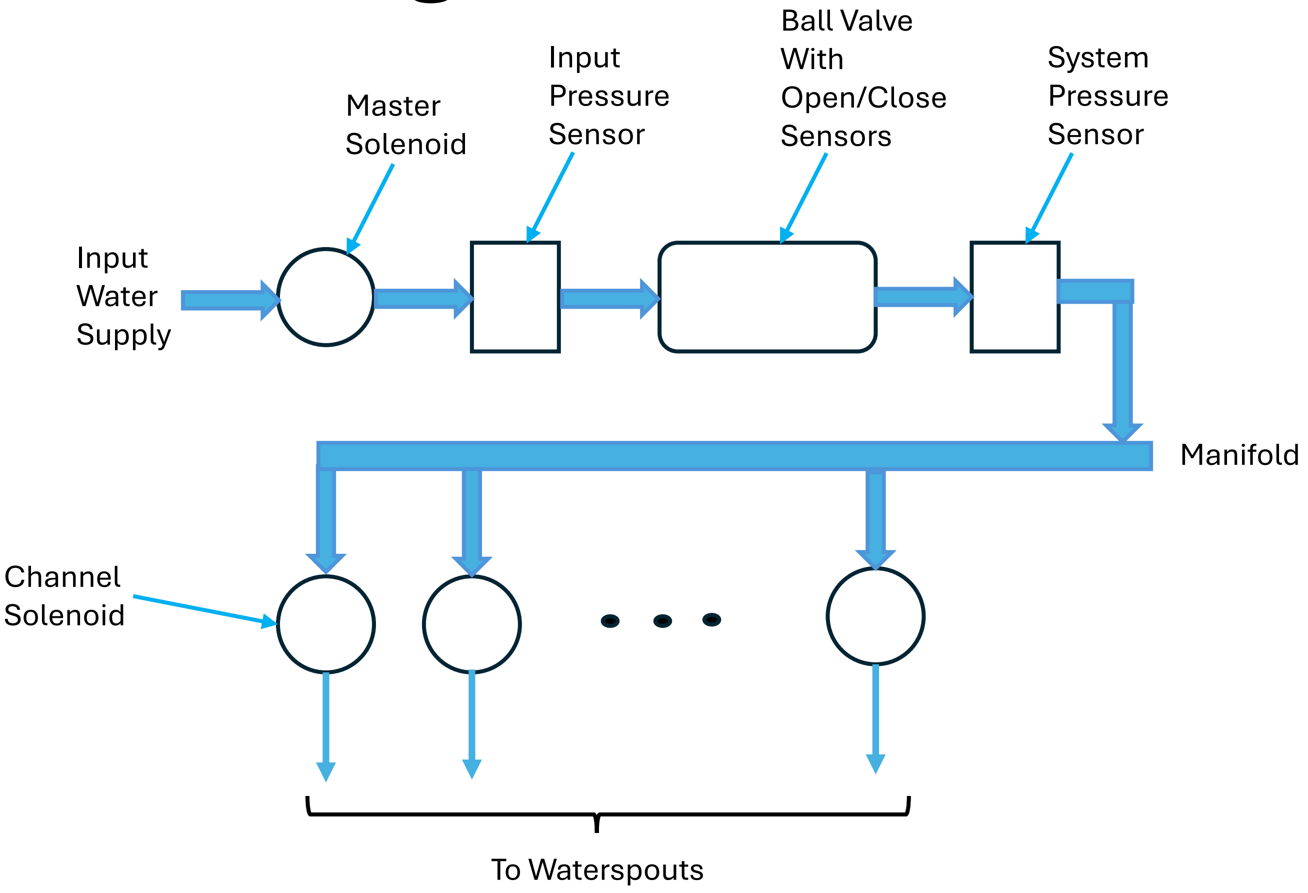


Power Supply in Wall under Window in Dinning Room

ON / OFF Control

- The toggle switch in the Dining room cabinet controls power to the vault.
- When the splash pad is NOT in use, the switch should be off. This will shut down all power to the vault.
- When the system is off, a master solenoid valve is shut off which shuts down all water into the system.
- When the system is on, a sump pump is activated if there is accumulated water in the vault. Therefore, turn the system on for a few minutes to clear water in the vault.

System Diagram



Notes:

- The Master Solenoid is controlled directly by the input power. If the power dies, water is shut off.
- The system pressure is controlled by the Ball Valve. This means all waterspouts are supplied the same pressure at the same time.
- All pipes from the supply to the channel solenoids are 1/2-inch PVC. Tubes are used after the channel solenoids.

Water Sensor

- If water accumulates (from a leak or rain) in the bottom of the vault, it has no-where to go.
- If accumulated water rises too high, it can destroy the machinery in the vault.
- To mitigate against this, a battery powered water sensor has been installed. If more than about 1 inch of water accumulates, the sensor will emit a loud twerping sound.
- To clear the water, enable the sump pump by turning on the system.

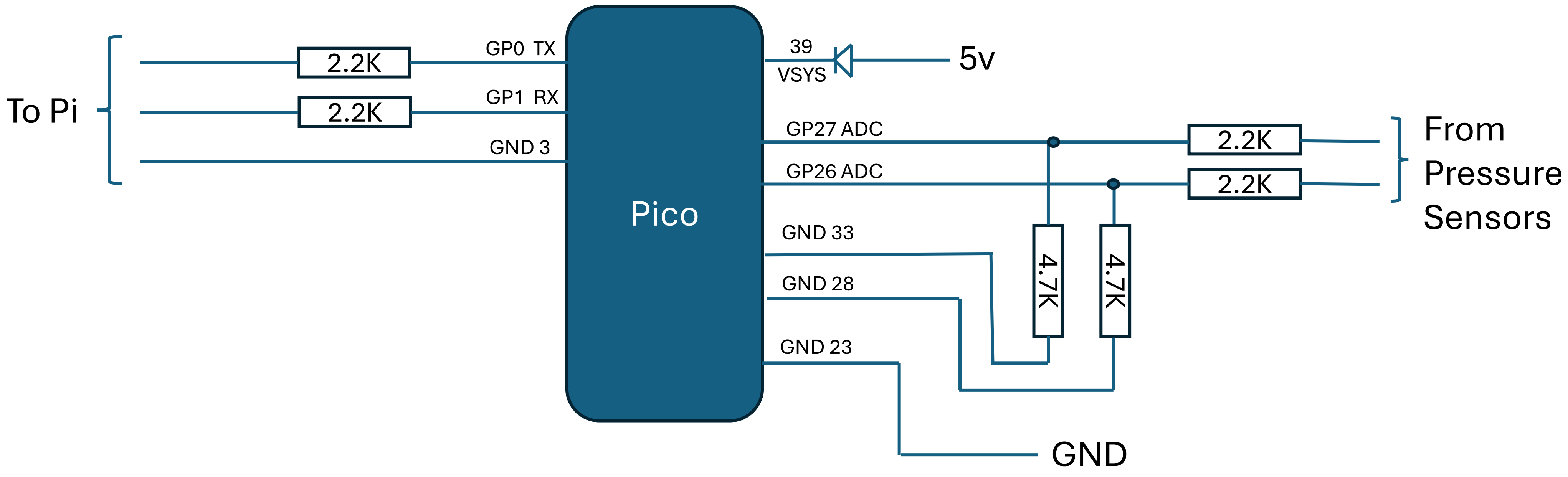
Raspberry Pi Setup

- Using Raspberry Pi 5
- Installed “Raspberry Pi OS with Desktop” – version 6.6, Debian 12
- Comes with Python 3.11.2
- Added 192.168.3.23 to DCHP Reservations in beach house router.
- Host name of pi is “SplashVault”
- Used “splash” and pw “hockeypuck” as main account in Raspberry.
- Enabled SSH. Did this by using ssh-keygen on Windows, and coping the .pub file to authorized_keys on pi.
- Able to SSH into pi with “ssh splash@192.168.3.23”
- Did more configurations, so now able to SSH from my office computer with just:

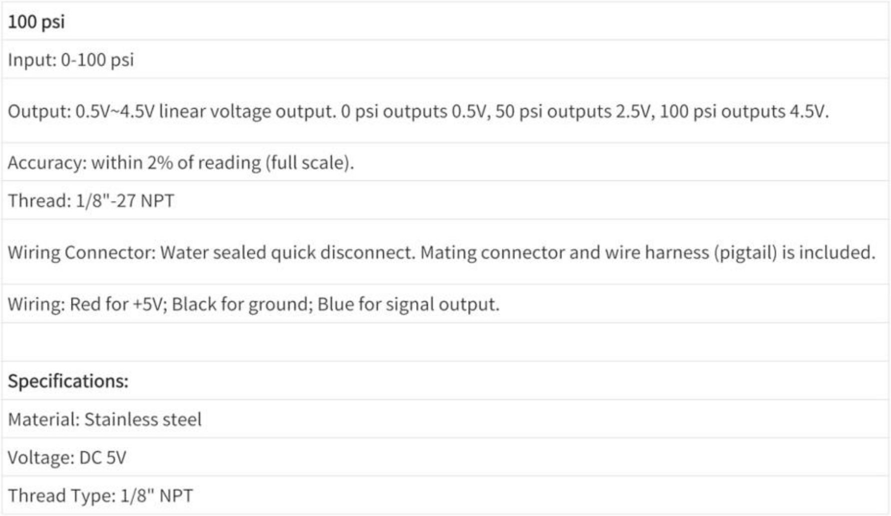
  ssh splash@SplashVault (no password necessary)

Using Pi Pico (RP2040) to Read Pressure Sensors

- Using Micro Python. Editor: Thorny.

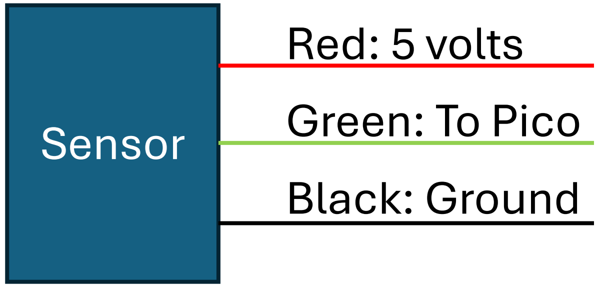


Pressure Sensor



100 psi
Input: 0-100 psi
Output: 0.5V~4.5V linear voltage output. 0 psi outputs 0.5V, 50 psi outputs 2.5V, 100 psi outputs 4.5V.
Accuracy: within 2% of reading (full scale).
Thread: 1/8"-27 NPT
Wiring Connector: Water sealed quick disconnect. Mating connector and wire harness (pigtail) is included.
Wiring: Red for +5V; Black for ground; Blue for signal output.
Specifications:
Material: Stainless steel
Voltage: DC 5V
Thread Type: 1/8" NPT

About $15 from Amazon

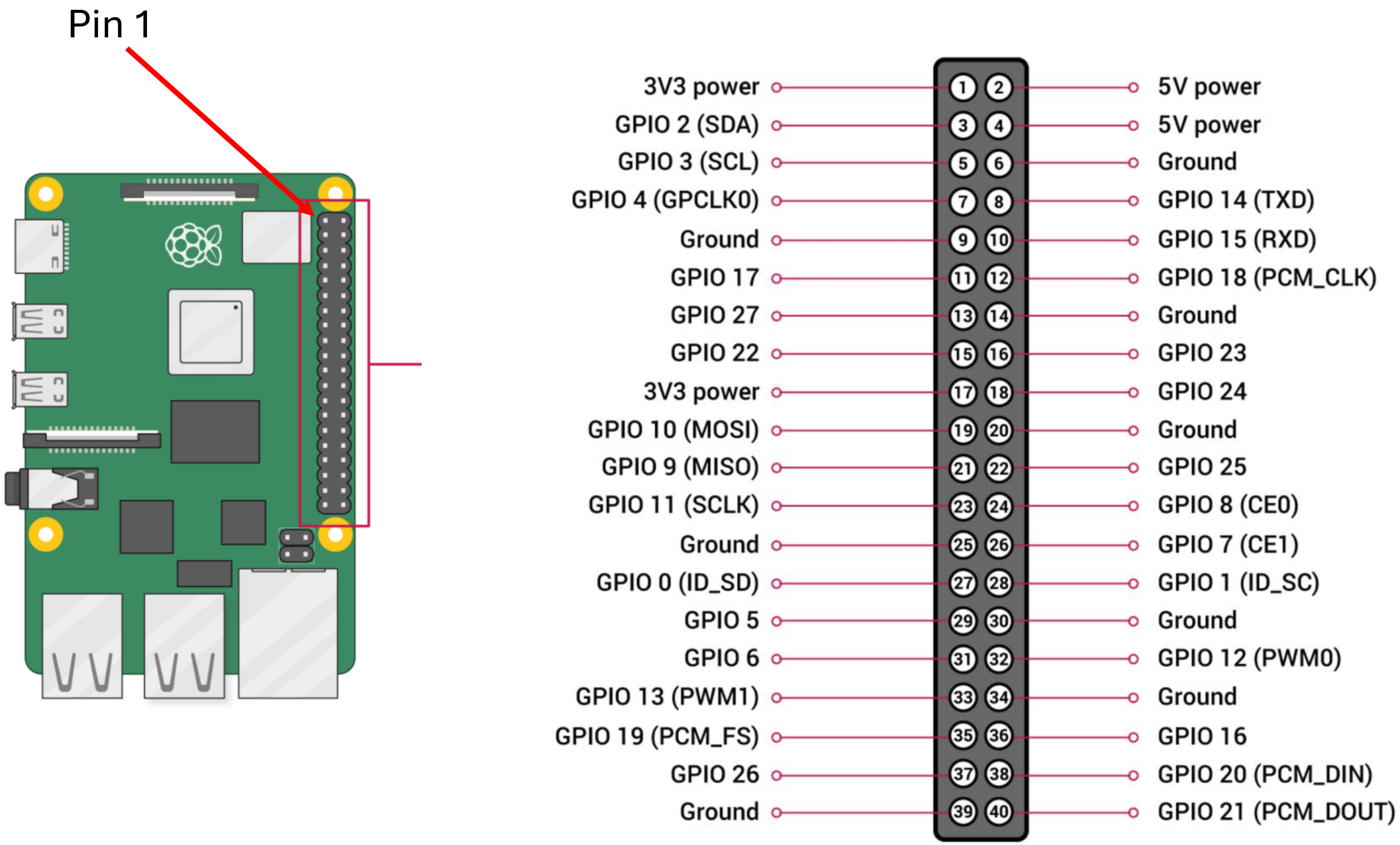


Code in Pico for Pressure Sensors

```
# For Reading Pressure in splash pad vault at the Beach House.
# DLB, Aug 2024

# The splash pad machine has two pressure sensors.  Each sensor
# reads PSI from 0 to 100, and outputs the reading in volts over
# the range 0.5 to 4.5 volts, where 0 psi is 0.5 volts, and 100
# psi is 4.5 volts.

# The signals from the pressure sensors are routed through a
# resistor network (2.2K/4.7K) to reduce the maximum voltage
# to about 3.1 volts, and then presented to GPIO Pins 26 and 27.

# The ADC is used to produce 16 bit readings.  These readings
# are encoded as follows: "xxxxx,yyyyy/n" where x and y are
# digits of the two readings.  This string is sent over a
# 921600 baud link to the host computer, each time the host
# requests the reading by sending a "?".
```

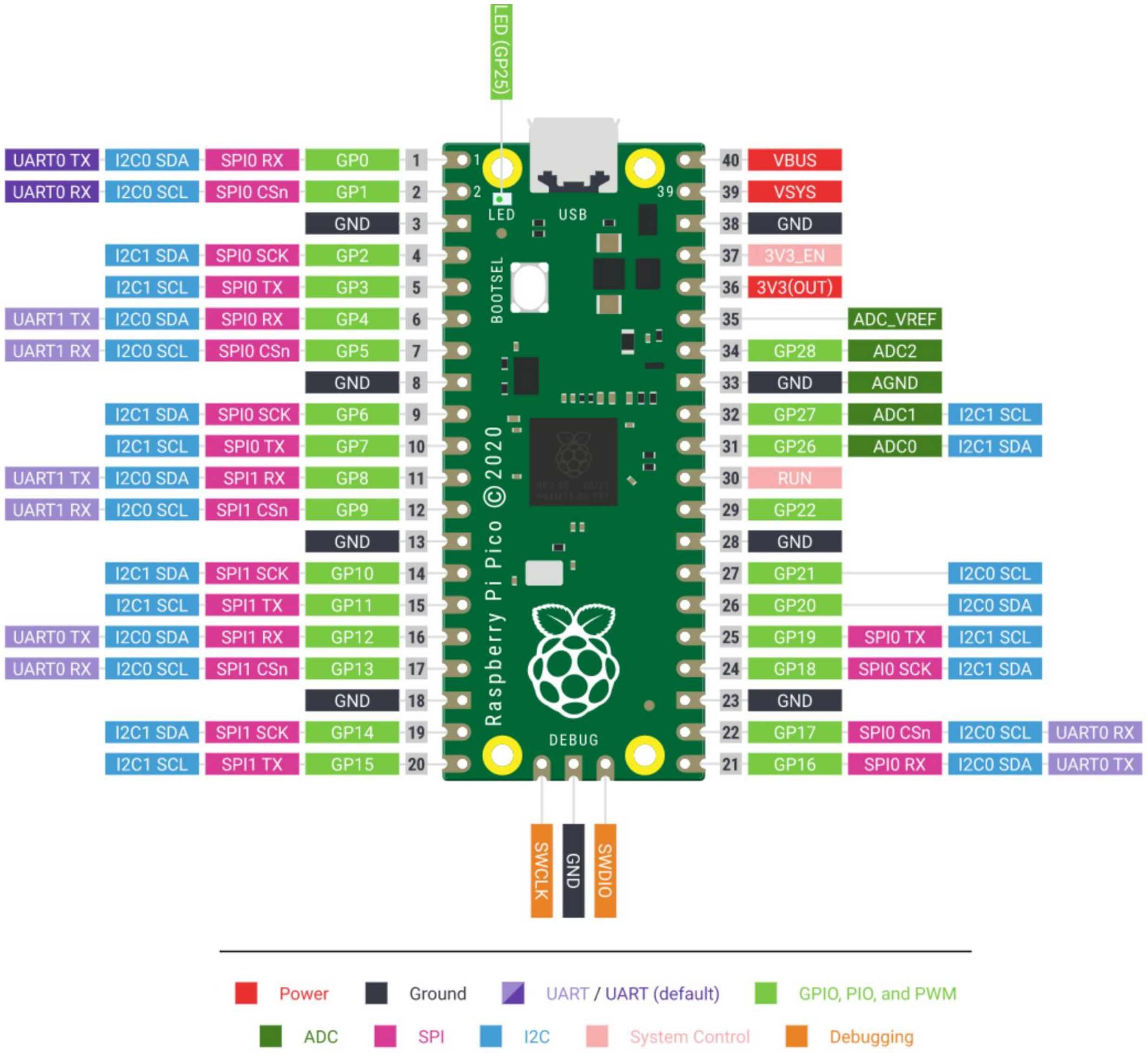
```
from machine import ADC
from machine import UART, Pin
from time import sleep
vpin1 = ADC(26)
vpin2 = ADC(27)
u = UART(0, baudrate=921600, tx=Pin(0), rx=Pin(1))
buf_input = bytearray(10)

v1 = 0
v2 = 0

def run():
    while True:
        sendit = False
        nwaiting = u.any()
        if nwaiting > 0:
            u.readinto(buf_input, nwaiting)
            #print("Received: ", buf_input)
            for i in range(nwaiting):
                if buf_input[i] == b'?'[0]:
                    sendit = True
                    break
        if sendit:
            s = b"%05d,%05d\n" % (v1, v2)
            u.write(s)
            #print("Sent: " + str(s))
        v1 = vpin1.read_u16()
        v2 = vpin2.read_u16()
```

Raspberry Pi 5   Pinout Diagram

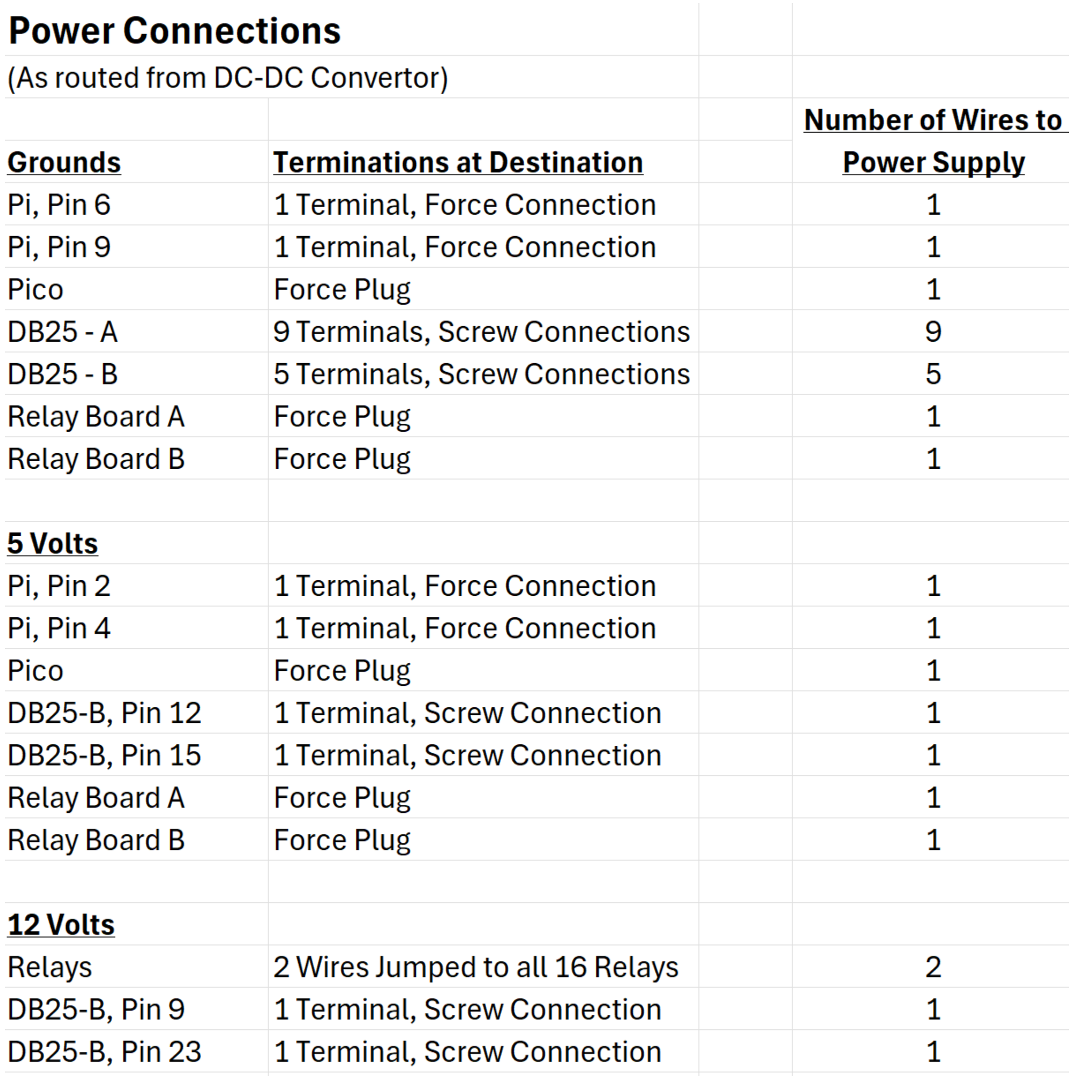


3V3 power	1	2	5V power
GPIO 2 (SDA)	3	4	5V power
GPIO 3 (SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (TXD)
Ground	9	10	GPIO 15 (RXD)
GPIO 17	11	12	GPIO 18 (PCM_CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3V3 power	17	18	GPIO 24
GPIO 10 (MOSI)	19	20	Ground
GPIO 9 (MISO)	21	22	GPIO 25
GPIO 11 (SCLK)	23	24	GPIO 8 (CE0)
Ground	25	26	GPIO 7 (CE1)
GPIO 0 (ID_SD)	27	28	GPIO 1 (ID_SC)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM_FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM_DIN)
Ground	39	40	GPIO 21 (PCM_DOUT)

Pico Pinout


LED (GP25)
UART0 TX I2C0 SDA SPI0 RX GP0 1
UART0 RX I2C0 SCL SPI0 CSn GP1 2
GND 3
I2C1 SDA SPI0 SCK GP2 4
I2C1 SCL SPI0 TX GP3 5
UART1 TX I2C0 SDA SPI0 RX GP4 6
UART1 RX I2C0 SCL SPI0 CSn GP5 7
GND 8
I2C1 SDA SPI0 SCK GP6 9
I2C1 SCL SPI0 TX GP7 10
UART1 TX I2C0 SDA SPI1 RX GP8 11
UART1 RX I2C0 SCL SPI1 CSn GP9 12
GND 13
I2C1 SDA SPI1 SCK GP10 14
I2C1 SCL SPI1 TX GP11 15
UART0 TX I2C0 SDA SPI1 RX GP12 16
UART0 RX I2C0 SCL SPI1 CSn GP13 17
GND 18
I2C1 SDA SPI1 SCK GP14 19
I2C1 SCL SPI1 TX GP15 20
40 VBUS
39 VSYS
38 GND
37 3V3_EN
36 3V3(OUT)
35 ADC_VREF
34 GP28 ADC2
33 GND AGND
32 GP27 ADC1 I2C1 SCL
31 GP26 ADC0 I2C1 SDA
30 RUN
29 GP22
28 GND
27 GP21 I2C0 SCL
26 GP20 I2C0 SDA
25 GP19 SPI0 TX I2C1 SCL
24 GP18 SPI0 SCK I2C1 SDA
23 GND
22 GP17 SPI0 CSn I2C0 SCL UART0 RX
21 GP16 SPI0 RX I2C0 SDA UART0 TX
1
2
39
LED
USB
BOOTSEL
Raspberry Pi Pico ©2020
DEBUG
SWCLK
GND
SWDIO
Power
Ground
UART / UART (default)
GPIO, PIO, and PWM
ADC
SPI
I2C
System Control
Debugging

Power Connections

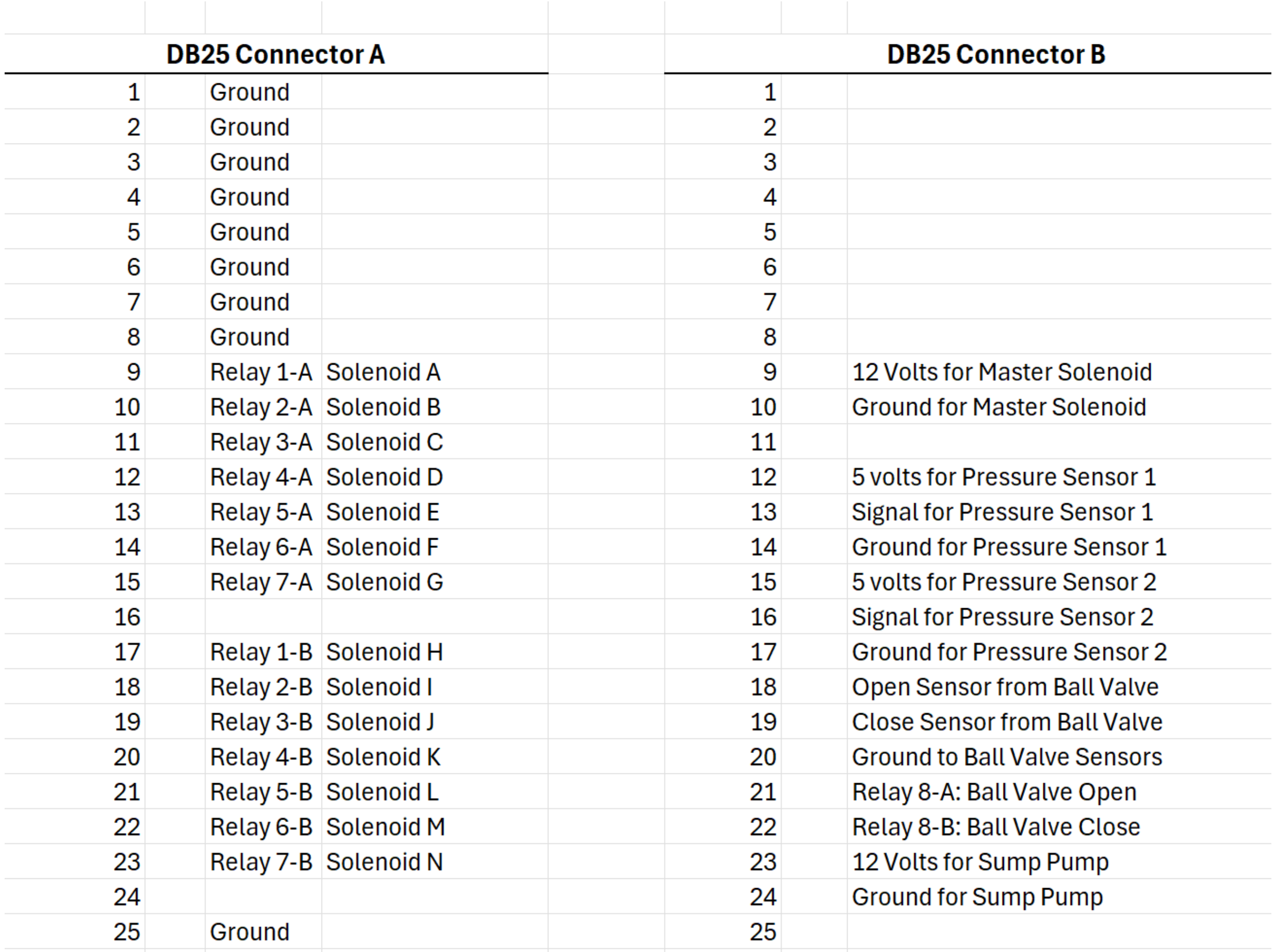


Power Connections

(As routed from DC-DC Convertor)

Grounds	**Terminations at Destination**	**Number of Wires to Power Supply**
Pi, Pin 6	1 Terminal, Force Connection	1
Pi, Pin 9	1 Terminal, Force Connection	1
Pico	Force Plug	1
DB25 - A	9 Terminals, Screw Connections	9
DB25 - B	5 Terminals, Screw Connections	5
Relay Board A	Force Plug	1
Relay Board B	Force Plug	1
5 Volts		
Pi, Pin 2	1 Terminal, Force Connection	1
Pi, Pin 4	1 Terminal, Force Connection	1
Pico	Force Plug	1
DB25-B, Pin 12	1 Terminal, Screw Connection	1
DB25-B, Pin 15	1 Terminal, Screw Connection	1
Relay Board A	Force Plug	1
Relay Board B	Force Plug	1
12 Volts		
Relays	2 Wires Jumped to all 16 Relays	2
DB25-B, Pin 9	1 Terminal, Screw Connection	1
DB25-B, Pin 23	1 Terminal, Screw Connection	1

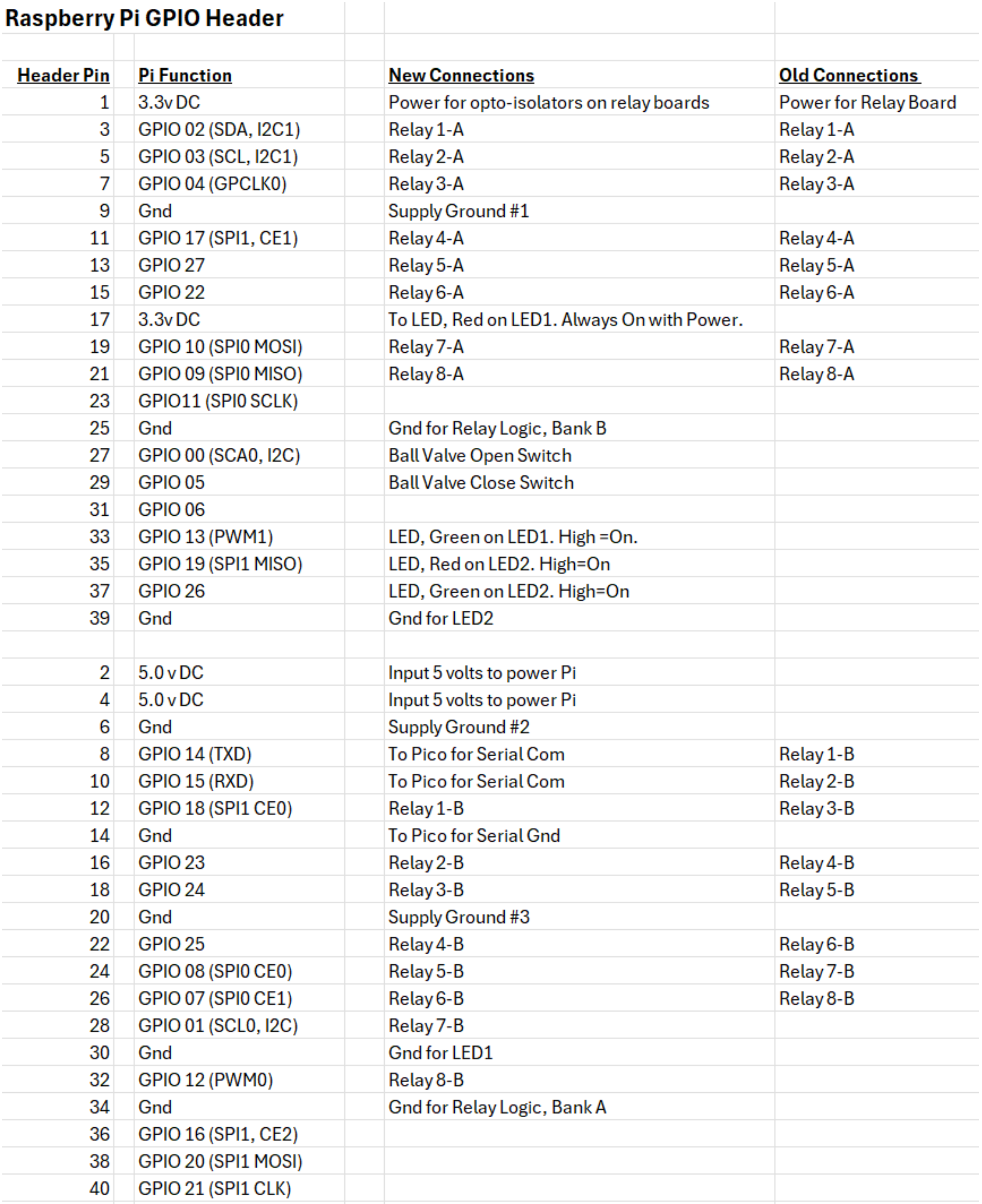
DB25 Connections



DB25 Connector A		
1	Ground	
2	Ground	
3	Ground	
4	Ground	
5	Ground	
6	Ground	
7	Ground	
8	Ground	
9	Relay 1-A	Solenoid A
10	Relay 2-A	Solenoid B
11	Relay 3-A	Solenoid C
12	Relay 4-A	Solenoid D
13	Relay 5-A	Solenoid E
14	Relay 6-A	Solenoid F
15	Relay 7-A	Solenoid G
16		
17	Relay 1-B	Solenoid H
18	Relay 2-B	Solenoid I
19	Relay 3-B	Solenoid J
20	Relay 4-B	Solenoid K
21	Relay 5-B	Solenoid L
22	Relay 6-B	Solenoid M
23	Relay 7-B	Solenoid N
24		
25	Ground	

DB25 Connector B	
1	
2	
3	
4	
5	
6	
7	
8	
9	12 Volts for Master Solenoid
10	Ground for Master Solenoid
11	
12	5 volts for Pressure Sensor 1
13	Signal for Pressure Sensor 1
14	Ground for Pressure Sensor 1
15	5 volts for Pressure Sensor 2
16	Signal for Pressure Sensor 2
17	Ground for Pressure Sensor 2
18	Open Sensor from Ball Valve
19	Close Sensor from Ball Valve
20	Ground to Ball Valve Sensors
21	Relay 8-A: Ball Valve Open
22	Relay 8-B: Ball Valve Close
23	12 Volts for Sump Pump
24	Ground for Sump Pump
25	

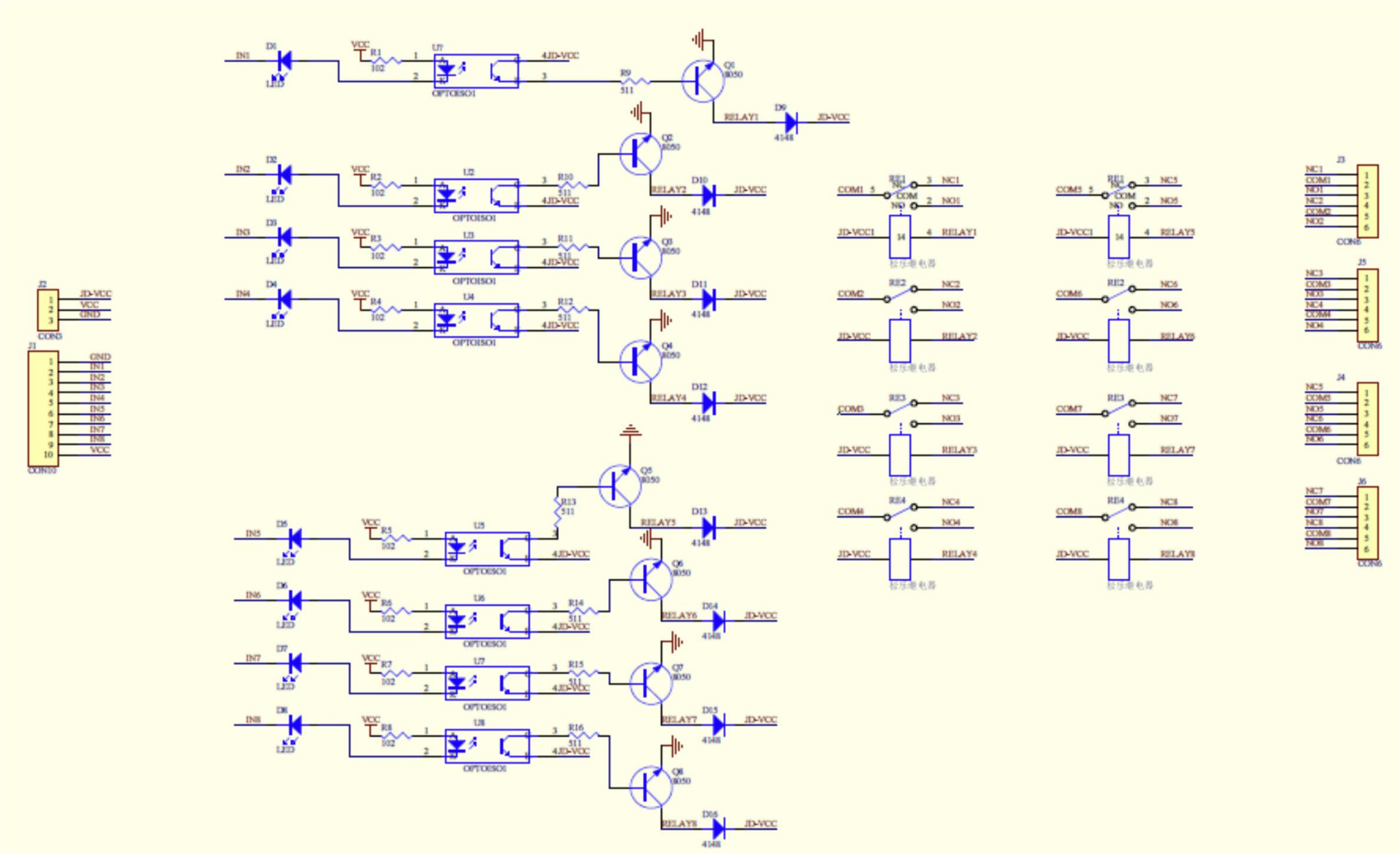
Raspberry Pi GPIO Connections



Raspberry Pi GPIO Header

Header Pin	Pi Function	New Connections	Old Connections
1	3.3v DC	Power for opto-isolators on relay boards	Power for Relay Board
3	GPIO 02 (SDA, I2C1)	Relay 1-A	Relay 1-A
5	GPIO 03 (SCL, I2C1)	Relay 2-A	Relay 2-A
7	GPIO 04 (GPCLK0)	Relay 3-A	Relay 3-A
9	Gnd	Supply Ground #1	
11	GPIO 17 (SPI1, CE1)	Relay 4-A	Relay 4-A
13	GPIO 27	Relay 5-A	Relay 5-A
15	GPIO 22	Relay 6-A	Relay 6-A
17	3.3v DC	To LED, Red on LED1. Always On with Power.	
19	GPIO 10 (SPI0 MOSI)	Relay 7-A	Relay 7-A
21	GPIO 09 (SPI0 MISO)	Relay 8-A	Relay 8-A
23	GPIO11 (SPI0 SCLK)		
25	Gnd	Gnd for Relay Logic, Bank B	
27	GPIO 00 (SCA0, I2C)	Ball Valve Open Switch	
29	GPIO 05	Ball Valve Close Switch	
31	GPIO 06		
33	GPIO 13 (PWM1)	LED, Green on LED1. High =On.	
35	GPIO 19 (SPI1 MISO)	LED, Red on LED2. High=On	
37	GPIO 26	LED, Green on LED2. High=On	
39	Gnd	Gnd for LED2	
2	5.0 v DC	Input 5 volts to power Pi	
4	5.0 v DC	Input 5 volts to power Pi	
6	Gnd	Supply Ground #2	
8	GPIO 14 (TXD)	To Pico for Serial Com	Relay 1-B
10	GPIO 15 (RXD)	To Pico for Serial Com	Relay 2-B
12	GPIO 18 (SPI1 CE0)	Relay 1-B	Relay 3-B
14	Gnd	To Pico for Serial Gnd	
16	GPIO 23	Relay 2-B	Relay 4-B
18	GPIO 24	Relay 3-B	Relay 5-B
20	Gnd	Supply Ground #3	
22	GPIO 25	Relay 4-B	Relay 6-B
24	GPIO 08 (SPI0 CE0)	Relay 5-B	Relay 7-B
26	GPIO 07 (SPI0 CE1)	Relay 6-B	Relay 8-B
28	GPIO 01 (SCL0, I2C)	Relay 7-B	
30	Gnd	Gnd for LED1	
32	GPIO 12 (PWM0)	Relay 8-B	
34	Gnd	Gnd for Relay Logic, Bank A	
36	GPIO 16 (SPI1, CE2)		
38	GPIO 20 (SPI1 MOSI)		
40	GPIO 21 (SPI1 CLK)		

Relay Module Schematic



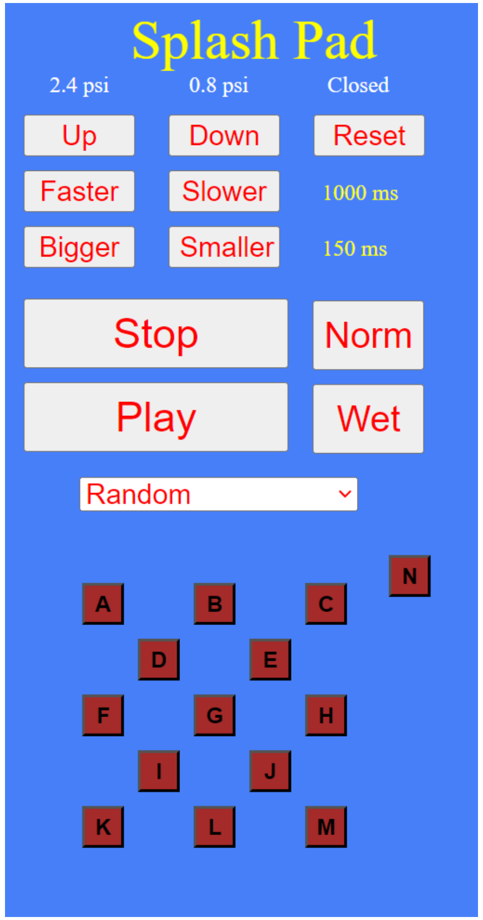
Power Conversion in Splash Controller Box

- Power Input is 12 volts at up to 30 Amps
- 12 Volts used for all Solenoids
- 5 Volts used for Pi, Pico, Relay Boards and Pressure Sensors
- 3.3 Volts (Produced by Pi) used for Logic side of Relay Boards
- Almost all current carrying wire used in the controller box is 20 AWG Silicon wire.
- 20 AWG wire is rated to 5 Amps.
- 20 AWG silicon wire has a resistance of 62.50 ohms per km, or about 0.01 ohms in a foot. At 30 Amps, Voltage drop would be 0.3 volts. Wire would need to dissipate 9 Watts. (Not good)

DC-DC convertor. Input: 12-24 volts.
Output: 5 Volts at 10 Amps

Spash Pad Control Website

Access this website at:
192.168.3.23:5000



Stops a running program

Runs the program indicated by the pulldown below.

Program selection pulldown

Reset completely closes the flow control valve.

Controls the overall water Flow.

Controls the cadence of the program.

Controls the duration of each squirt

Norm sets up the flow, cadence and duration for typical operation.

Wet sets up the flow, cadence and duration for wetter operation.

Allows individual control of each spout. (Shuts off any running program.)

Note: the flow, cadence and duration controls do not work with all programs. That is, some programs take over control of these parameters.

Note: The status numbers only update on a page refresh.