SprinklerPI

(Design)

Jeremiah Mahler
EECE 490B, CSU Chico
February 6, 2014

Contents

1 Architectural Design

At the highest level the SprinklerPI system creates a web interface for controlling sprinkler valves. Figure 1 gives an overview.

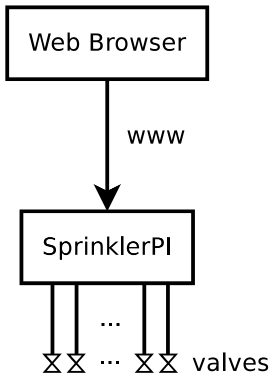

Figure 1: Overview of SprinklerPI in system.

There are many subcomponents of the SprinklerPI system. The power supply must take 110 volts AC from a residential outlet The power supply must take 110 volts AC from a residential outlet and convert it to 24 volts AC and 5 volts DC. The 24 volts AC is needed to drive the sprinkler valves. The 5 volts DC is needed for the Linux computer and the digital logic (control). The digital logic (control) must take accept a command over SPI and turn on one of the sprinkler valves or turn them all off. The driver must take a digital logic input and drive the appropriate sprinkler valve using 24 volts AC. The web based user interface must allow the timer schedules to be set and for the valves to be manually operated. Figure 2 gives an overview of the major components inside the SprinklerPI system.

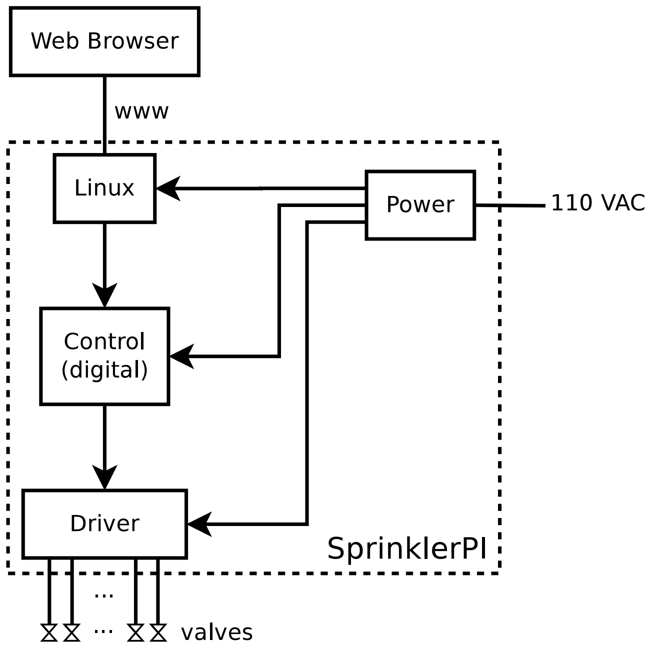

Figure 2: Overview of main components inside the SprinklerPI system.

2 Hardware Design

There are three main hardware components in this system. First is the power supply. It takes 110 volts AC and outputs both 5 volts DC and 24 volts AC. The sprinkler valves require 24 volts AC. The Linux computer and digital logic require 5 volts DC. Second is the Linux computer which is a standard RasberryPI model B. And third is the Control Driver which contains all the circuitry needed to accept commands using SPI from the RasberryPI and to drive up to eight individual sprinkler valves. Additional Control Drivers can be added to drive additional sprinkler valves. Each individual Control Driver supports eight valves. To keep the system modular each of these modules will be on a separate PCB. Figure 3 gives an overview of the hardware in the system.

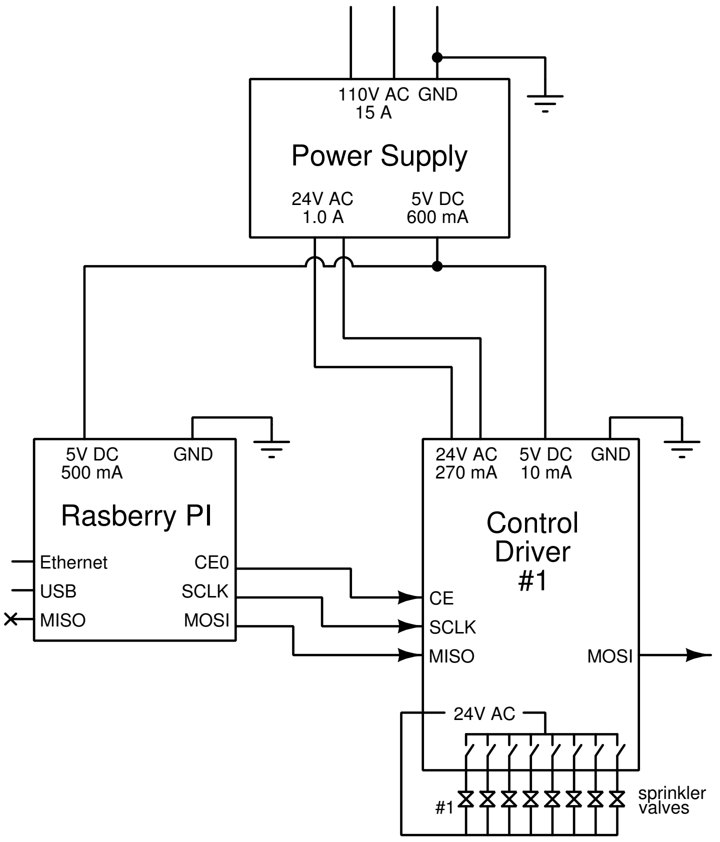

Figure 3: Overview of hardware in system. Only one control/driver is shown, additional control/drivers can be added up to a maximum of three. Expansion is limited by the power supply.

2.1 State Flow Diagrams

The primary module of interest with regard to state flow is the Control Driver (Figure 3). It controls the sprinkler valves based on commands it receives from the Linux computer (RasberryPI) over SPI.

The Control Driver is always either driving one of the eight valves or has all of the valves off. The only time this changes is when a new command is sent.

During power on all the valves must remain in the off state. If this were not required, and the system entered some random state during power on, a valve could be inadvertently turned on. And since there is no watchdog mechanism this valve would remain on until the next command was received. Figure 4 shows the state machine of the Control Driver circuitry in hardware.

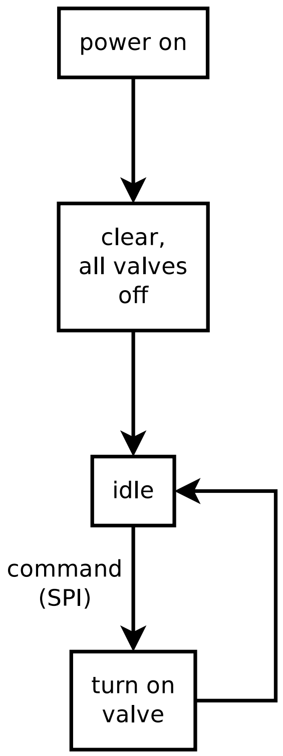

Figure 4: State diagram of hardware control circuitry. During power on it is important that the valves remain off. In the idle state it waits for a command to turn on a valve or turn off all valves.

The hardware state is determined by the control circuitry described in Section 2.2.1.

2.2 Schematics

2.2.1 Control

Before a sprinkler valve can be driven some control logic is necessary between the RasberryPI and the driver (Section 2.2.3). Most residential sprinkler systems are capable of driving eight circuits. And only one of these circuits is active at any one time [1]. This controller is similar except it also supports expansion by groups of eight circuits.

This design uses SPI and a minimum of 8-bits is sent per message. Each control group requires 4-bits. So for one to two control groups 8-bits are required. For three to four control groups 16-bits are required, and so on. Figure 5 shows the message format.

Each 4-bit control message contains a single enable bit. This design was chosen to simplify the clear of the decoder (Figure 6).

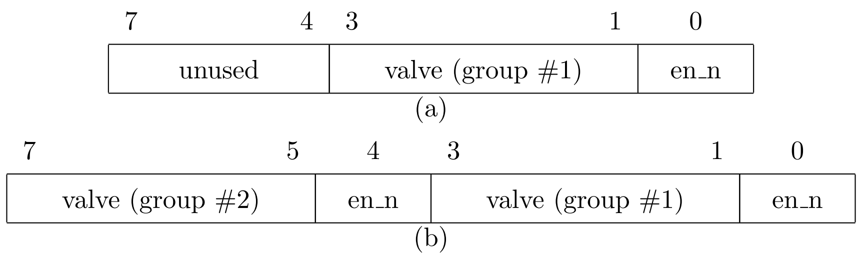

Figure 5: SPI message format for one group (a) and two groups (b). Each group represents eight valves so (b) would support 16 valves.

Translating the message received over SPI to turn on a sprinkler valve requires a small amount of discrete logic. A shift register is needed to take the bits as inputs. A decoder is needed to convert shifted in number to a single bit output. In this case a 74HC238 3-8 decoder is used since it provides non-inverting outputs. An output high will turn on a valve. Figure 6 shows the control schematic.

On power up it is important that the valves remain off. However the shift register chip (74HC194) does not guarantee the output state during power up. To resolve this issue a simple RC circuit is used to to hold the chip in the clear state for several seconds after power on (Figure 6).

[1] If multiple circuits were active this could lower the water pressure resulting in inconsistent watering amounts.

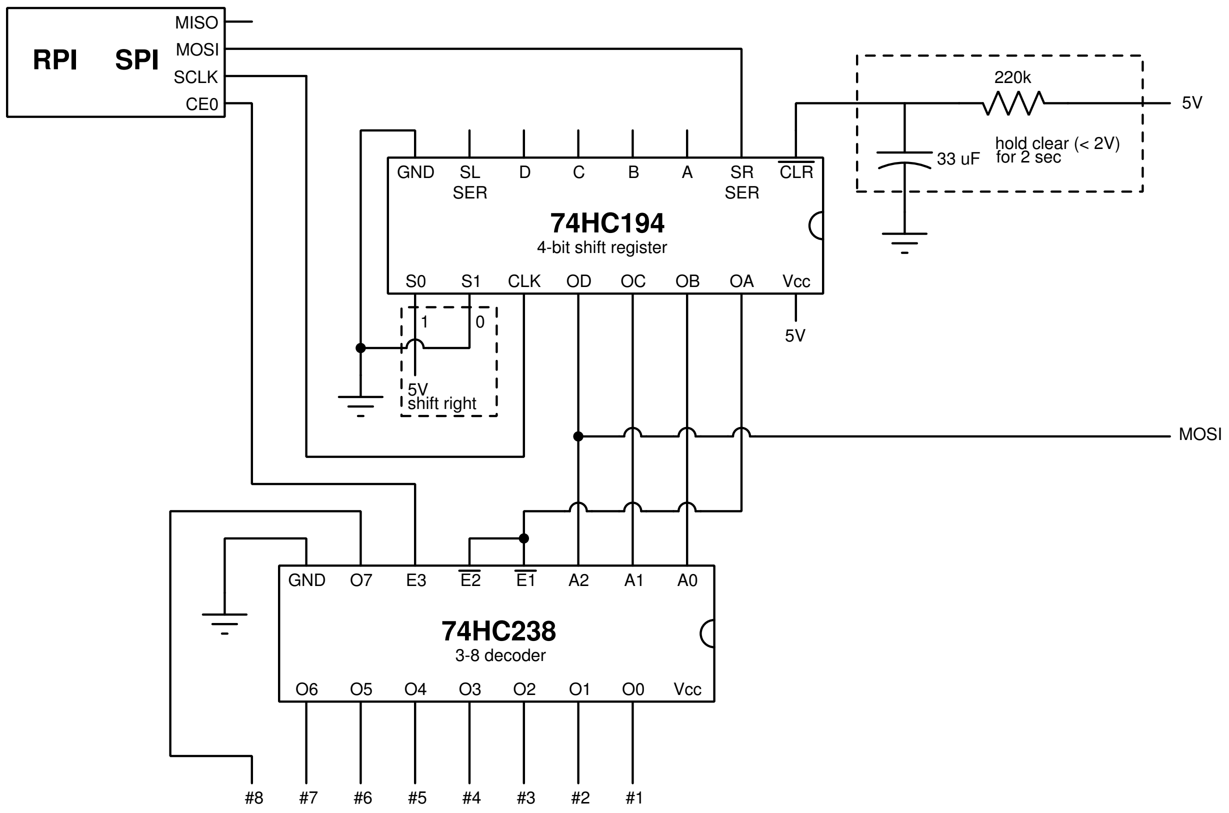

Figure 6: Sprinkler valve control logic. Sprinkler valve number is input using SPI from the RasberryPI in to the shift register. The number is decoded and inverted to produce a single high signal for one valve. All outputs are disabled when OA is clear.

2.2.2 Expansion

The RasberryPI uses SPI to communicate with the control logic which which drives the sprinkler valve. Each group can control up to eight valves with one valve being on at a time. A daisy chain arrangement can be constructed to allow SPI to communicate with more than one group as shown in Figure 7.

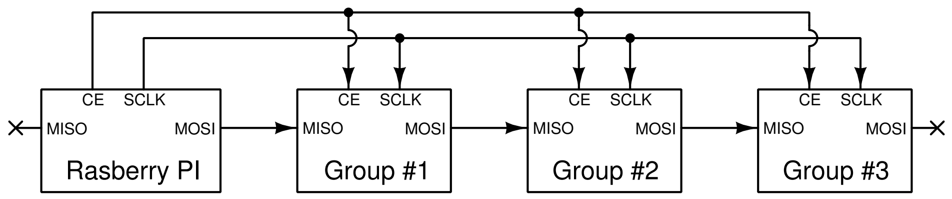

Figure 7: Daisy chain SPI configuration to for multiple control groups.

With the current power supply design it is limited to three control groups as shown in Figure 8. Each group can control eight valves with one being on at a time. This makes a total of 24 valves where three can be on at once.

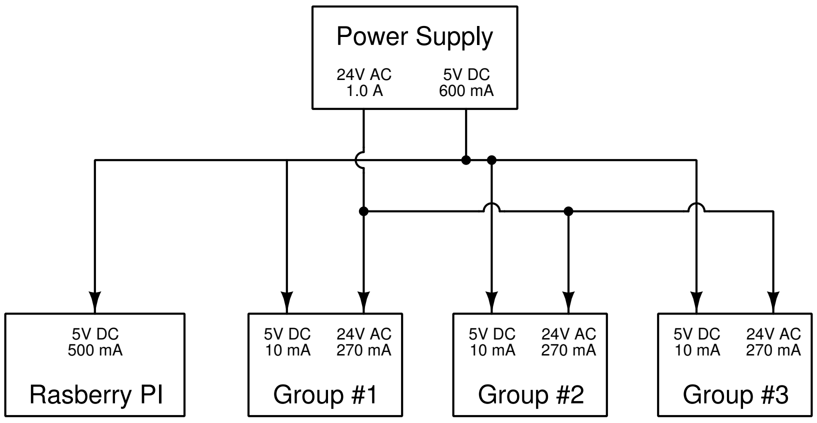

Figure 8: With a power supply capable of 1 amp at 24 volts AC and 600 mA at 5 volts DC a maximum of three control groups and one Rasberry PI can be driven.

2.2.3 Driver

The control circuit (Section 2.2.1) provides a digital output signal in the range of 0 to 5 volts DC. This signal is inadequate for driving the sprinkler valve which requires 24 volts AC. The driver circuit discussed here is used to apply 24 volts AC signal when triggered by a 5 volts DC signal.

To turn the sprinkler valves on requires 24 VAC at approximately 270 mA. If it is assumed that during steady state both the solenoid and the triac have negligible resistance (Figure 9), a 10 Ω resistor can be used to limit the current to 270 mA. At 1 watt the resistor supports the worst case amperage with a margin over 10%.

$$\begin{aligned} P &= I^2R \\ &= (270 \times 10^{-3})^2 \cdot 10 \\ &= 0.729 \quad [\mathrm{W}] \end{aligned}$$

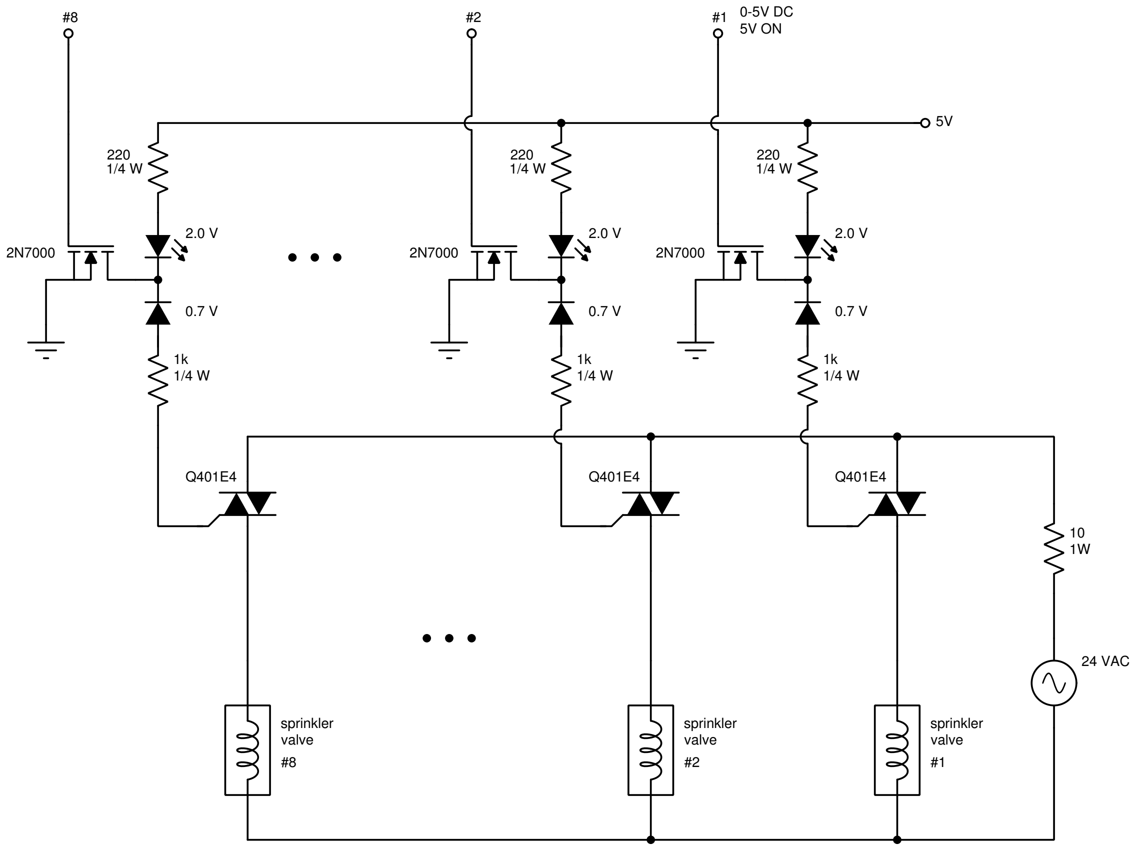

Figure 9: Sprinkler valve driver circuits. Not all branches are shown since they are identical. LEDs are used to indicate when the valve is on/off. It is assumed that only one valve will be on at a time.

The control signal to the triac (Q401E4) is very sensitive. The tinyest current will cause it to conduct. If it is driven by a 0-5V signal it will conduct not only with a 5V signal but also with 0V signal. To overcome this issue a diode was used to ensure no current flows when it is reverse biased. The FET (2N7000) is used to switch the triac while it also serves to limit the control current of a 0-5 volt signal to less than 0.5 mA. This places the control current well within the limits of CMOS logic as required by the control chips (Section 2.2.1).

2.2.4 Power Supply

The power supply must provide two different voltages: 24 volts AC and 5 volts DC. The sprinkler valves require 24 volts AC at 270 mA. The Linux computer and other digital logic require 5 volts DC with a maximum draw of 600 mA [2]. These voltages must be created from a 110 volt AC input.

The 24 volts AC can be achieved using a split pole transformer with two 12 volt AC outputs. By bridging the center of these two 12 volt AC outputs a combined 24 volt AC output is available.

A switching voltage regulator, the MC34063A, was chosen for the 5 volt supply. It has a greater efficiency than a linear regulator such as the LM7805.

The circuit which achieves all of these design requirements is shown in Figure 10.

[2]The current draw of the RasberryPI was determined experimentally to be approximately 500 mA.

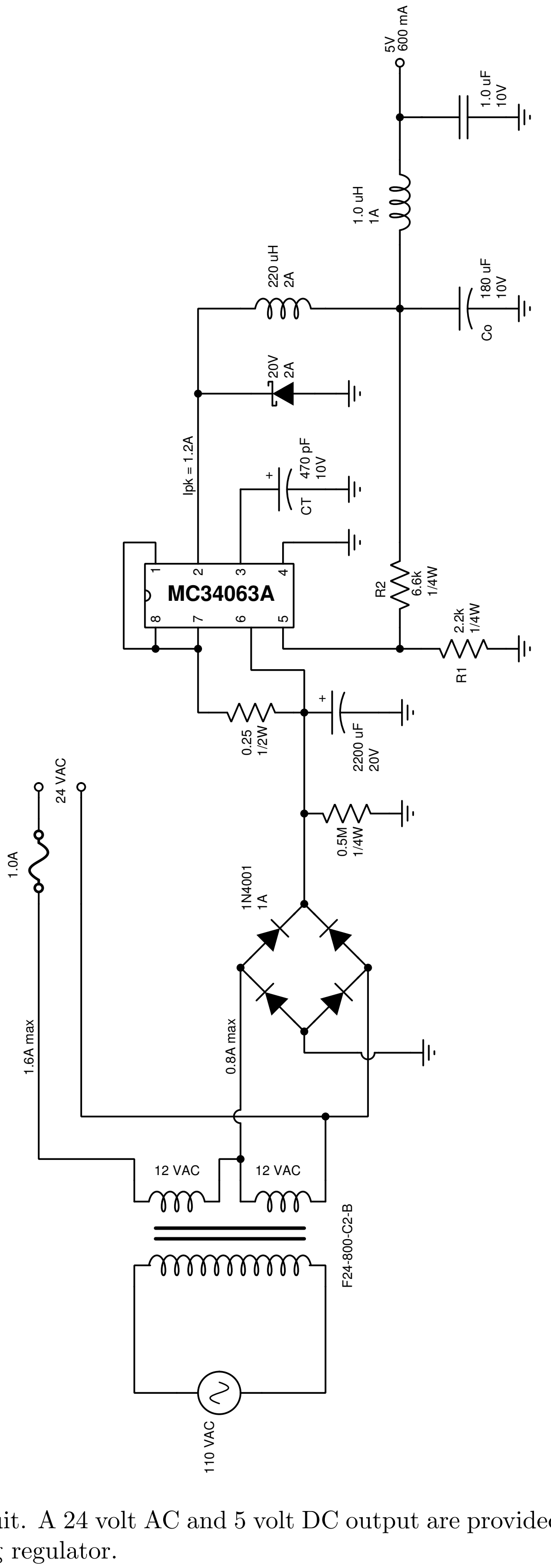

Figure 10: Power supply circuit. A 24 volt AC and 5 volt DC output are provided. The 5 volt DC is provided by the MC340363A switching regulator.

2.3 Timing Diagrams

Figure 11 is a diagram of a message being sent using SPI from the Linux computer to the control module. Refer to Section 2.2.1 for a description of the meaning of these messages.

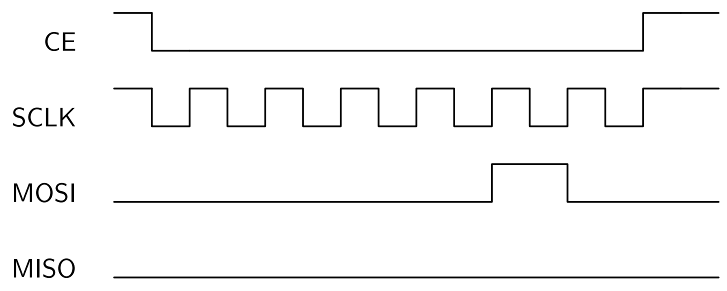

Figure 11: Timing diagram of a message sent using SPI from the RasberryPI to a control module. A value from MOSI is latched on the rising edge of SCLK. In this case the value 2 is transferred.

2.4 Theory of Operation

The operation of the hardware as a whole provides the ability to turn a valve on or to turn them all off. One valve can be on at a time for a single Control Driver module. Commands are written over SPI from the Linux computer (RasberryPI).

The user interface, described in Section 3, will establish a friendly user interface on top of this simple hardware interface.

3 Software Design

The software creates a user interface for controlling and scheduling the sprinkler valves. Figure 12 gives an overview of the system.

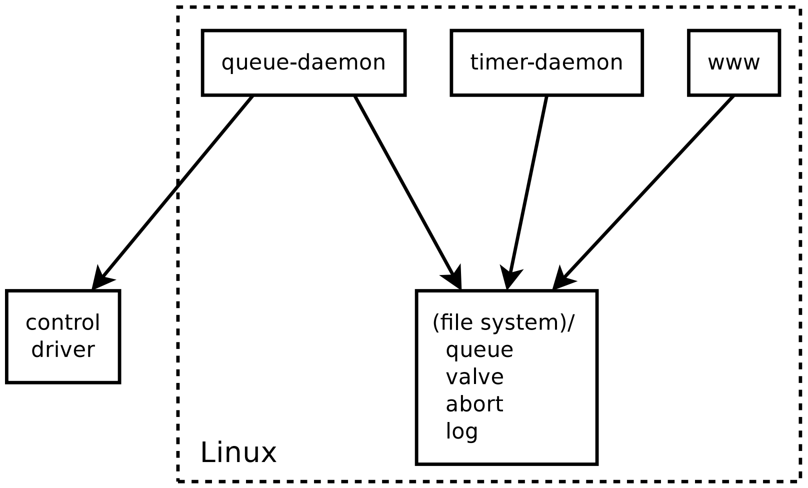

Figure 12: Overview of software in the sprinkler control system. The web server (www) communicates with the queue daemon using a set of files. The queue daemon monitors the queue and controls the valves accordingly. The timer daemon adds jobs to the queue according to the timer schedule.

3.1 File System

Interprocess communication between the queue daemon, the web server and the timer daemon is accomplished using files. The following is a listing of the files that are present.

```
sprinklerpi/
  queue
  valve
  clear
  log
  timer/
    valve1
     ...
    valve8
```

- `queue` - Queue of jobs to execute.
- `valve` - Current valve number that is running. Zero if none are running.
- `clear` - When true the queue daemon clears the queue. It is reset to false after queue is cleared.
- `log` - Log of operations queue-daemon has performed.
- `timer/valve*` - Schedule of when each valve should run.

3.2 Queue Daemon

The queue daemon monitors the queue file and executes the command. An example command might be "run valve 3 for 5 minutes". Figure 13 gives a flow chart of its operation.

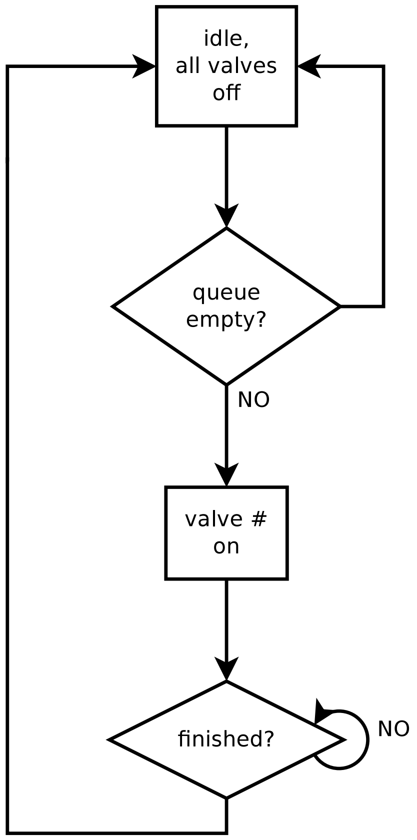

Figure 13: Flow chart of the queue daemon operation.

3.3 Timer Daemon

The timer daemon monitors the valve schedule and adds jobs to the queue when they are ready to be run. Figure 14 gives a flowchart of its operation.

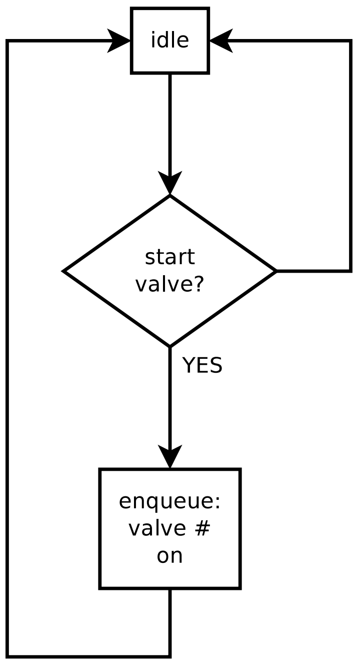

Figure 14: Flow chart of the timer daemon operation.

4 Future Directions

4.1 Automated Hardware Testing

Currently the operation of the system can be manually verified by a user. There are LED indicators to show when a valve is on/off. And programs can be written to operate all the valves. It can also be verified by a user that the sprinkler valves operate when connected to an actual sprinkler system. However this process is far too slow and costly for testing this system if it was manufactured.

One solution is to create an additional module which can place a load on the driver circuit for each of the eight valves. And by measuring the voltage/current it can be determined if the correct circuit is being actuated and if its rated current is being drawn.

And programs can be written to control the system and verify its operation using this test module. The entire system could be tested and verified in a few minutes.

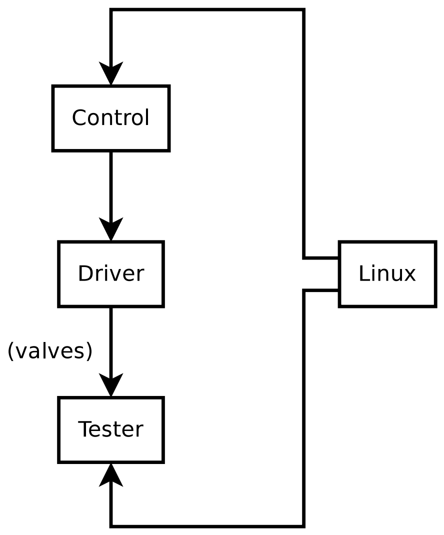

Figure 15: Automated testing system. The tester is connected in place of the actual sprinkler valves.

4.2 CAN Bus

The SPI bus will quickly become unwieldy to use with many controllers. In a daisy chain configuration each controller requires eight bits. As an example, with ten controllers, an eighty bit command would have to be sent. The CAN bus is much more capable it would be a good alternative.

4.3 Battery Backup

Most residential sprinkler systems have a 9 volt battery to keep power applied to prevent the schedules from being lost if the power briefly goes out. This system does not have this same need since data is written to flash on the Linux computer is kept during a power outage.

The battery backup idea could be expanded in this case to make the system more robust. If the system could continue watering even when the power is out this could be a valuable feature beyond a standard sprinkler system.

If a re-chargeable battery was used an appropriate charging circuit would need to be constructed.

Since the valves require 24 volts AC this would need to be generated from DC as well.

Glossary

FIFO First in first out.

GPIO General purpose input/output.

MOSI (SPI) master out slave in.

RFID Radio frequency identification.

SPI Serial peripheral interface.

References

Foundation, Rasberry PI. *Rasberry PI.* [Online; accessed 18-August-2013]. 2013. URL: `http://www.rasberrypi.org`.