Directed Study Manual

Dual Pump Water Distribution System

Student: Mauro Pérez

Advisor: Gary Holness

Institution: Clark University

Contents

Chapter 1

Introduction

This directed study focuses on the design and implementation of a command-driven IoT water transfer system. The project uses an Arduino-based controller, WiFi communication, flow measurement, distance sensing, and pump control to move a requested amount of water between two containers. The purpose of the project was to develop a user-controlled transfer module on top of the automatic water dispenser. A user can request a specific transfer from one container to the other; making the system a lot more flexible and ready for real-world use cases. A user is able to choose the direction and amount for transfer, while the sensor help evening out the amounts between tanks.

The software is split into two main parts, the Arduino which handles the physical system running like a micro controller. The Arduino connects to the WiFi, receives the commands, reads the sensors, controls the pumps, and does calculations. The Ocaml is simply the controller program, that interprets the commands.

Using a TCP protocal the OCaml uses simple text. The Arduino program follows a finite state machine structure with the states `INIT`, `GET_REQUEST`, `PARSE_REQUEST`, `SENSE`, `THINK`, and `ACT`. This structure keeps the program organized by separating network communication, sensor measurement, decision making, and pump control.

The OCaml program acts as the user-side controller. It provides a command-line

interface where the user can type commands such as `status`, `measure`, `stop`, or `transfer A B 1.5`. The OCaml program then converts those commands into the exact line-based protocol expected by the Arduino, such as `transfer,A,B,1.500`. Each command is sent through a TCP socket to the Arduino IP address and port. The reason for this project was to create awareness on how many of the Caribbean water systems are old, damaged, and unreliable. This project aims to better help those in need, and be better prepared to manage their water reserves. This project can have many use cases, from using it to distribute from the "cisterna" to other tanks, to use it as an irrigation system, or to pump water from one place to another.

Chapter 2

Software and Hardware

The Arduino acts as the device-side controller. It connects to WiFi, opens a TCP server on port `4080`, waits for incoming commands, performs the requested action, and returns a short text response. The Arduino program follows a finite state machine structure with the states `INIT`, `GET_REQUEST`, `PARSE_REQUEST`, `SENSE`, `THINK`, and `ACT`. This structure keeps the program organized by separating network communication, sensor measurement, decision making, and pump control. The Ocaml is the user side part of the control providing a command line interface that is able to transform the classes of the Arduino into commands that make it easier to execute; `status`, `measure`, `stop`, or `transfer A B 1.5`. The OCaml program then converts those commands into the exact line-based protocol expected by the Arduino, such as `transfer,A,B,1.500`. Each of these commands is sent through the TCP socket to the Arduino IP and port. Each pump has a separate task:

- `A_TO_B`: Pump 1 moves water from Tank A to Tank B.
- `B_TO_A`: Pump 2 moves water from Tank B to Tank A.

Using the H-bride pins, you are able to control which pump receives the electric signal to activate. The use of two types of sensor are used, one senor is a flow sensor that measures the amount of water that moves through the system, and the ultrasonic sensor which measures the amount of water in each tank, and as well serves as a way to activate one of the pumps if

the water is to close. The flow sensor is connected to an interrupt pin. Each pulse increases a counter called `flowPulses`. The software uses the calibration value:

$$\texttt{PULSES\_PER\_LITER} = 5880$$

This value comes from the flow sensor relationship:

$$F(Hz) = 98 \times Q(L/min)$$

Since there are 60 seconds in a minute, the pulse estimate becomes:

$$98 \times 60 = 5880 \text{ pulses per liter}$$

During a transfer, the firmware records the pulse count at the start of the cycle. It then compares the current pulse count to the starting pulse count. The difference is converted into liters:

$$\text{dispensed liters} = \frac{\text{current pulses} - \text{starting pulses}}{5880}$$

When the calculated dispensed volume reaches the requested target volume, the firmware stops the pump and resets the transfer state.

2.1 Sense-Think-Act Model

The Arduino State Machine is something learned in my IoT class, it is a simpler; below is a list that helps understand what each state entails:

`INIT` Connects to WiFi, prints network information, and starts the TCP server.

`GET_REQUEST` Checks whether a client has connected and sent a command.

`PARSE_REQUEST` Reads the command and calls the correct command handler.

`SENSE` Reads the ultrasonic sensor and updates the current transfer volume using the flow sensor pulse count.

`THINK` Decides whether the active transfer has reached the target volume.

`ACT` Stops the pumps when the target volume has been reached.

This **SENSE**, **THINK**, **ACT** model used for the transfer of water is very important as it waits for specific queues and triggers instead of it being an idle state machine, it is a dynamic one that is constantly weary of changes in its own system. This model allows for constant sensing and interpretation of information, and given appropriate commands you can easily control all aspects of the system.

2.2 System Architecture

The System architecture is made up of the OCaml controller, the Arduino firmware, WiFi network, and the hardware. The Ocaml runs the user-interface side of the system with ease if use and access through the development of the script, with each word having a class that it calls to the device controlling Arduino. Each command is transformed into a command type in OCaml, then Ocaml converts it into the message the Arduino expects. An example of a Ocaml command converted into one that the Arduino expects:

```
transfer A B 1.5
```

This command tells the system to transfer 1.5 liters of water from Tank A to Tank B. The OCaml controller checks the command, identifies the source tank, destination tank, and requested volume, then converts it into the Arduino protocol format:

```
transfer,A,B,1.500
```

When that command is executed the Ocaml controller opens a TCP socket connection to the Arduino using its IP address and port. This command is sent over WiFi as a text

message the Arduino can parse this command, select which direction the water travels to, make sure the sensor is still reading and execute. The Arduino continues running through the Sense-Think-Act loop while the transfer is active. During the `SENSE` stage, it reads the sensor values and updates the amount of water transferred. During the `THINK` stage, it compares the transferred amount to the requested target. During the `ACT` stage, it stops the pump once the target volume has been reached.

The architecture can be understood as a chain of responsibility:

1. The user decides what transfer should happen.
2. The OCaml controller translates the user command into Arduino protocol.
3. The TCP connection sends the command over WiFi.
4. The Arduino receives and parses the command.
5. The Arduino chooses the correct pump direction.
6. The flow sensor measures the water movement.
7. The Arduino calculates the transferred volume.
8. The Arduino stops the pump when the target volume is reached.
9. The Arduino returns a response to the controller.

2.3 Command and Response Flow

Each command from the OCaml interface to the Arduino is intentionally simple, which allows for versatile use of the system. Below is a diragram model on how the system operates and communicates for ease of understanding:

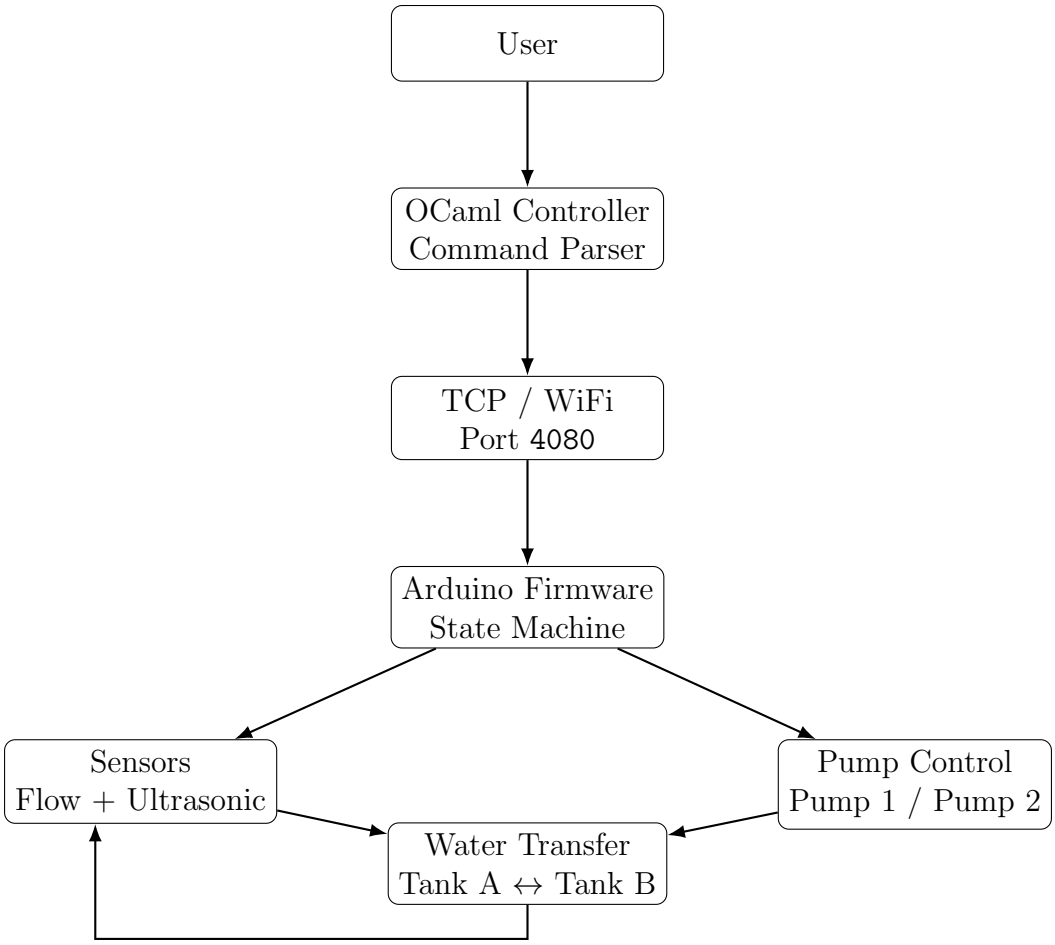

Figure 2.1: System architecture for the dual-pump water transfer system

The architecture creates a separation between the user interface and the hardware control. The OCaml program handles user input and communication, while the Arduino handles the real-time sensor readings and pump behavior. This separation makes the system easier to test and tailor the systems architecture to your specific needs and designs.

Chapter 3

User Manual

This section will contain a guide for the ease of use and alterations. Photos of the wiring will be included to help, however the wiring itself should be pretty straightforward, this handle the operation side along with some hardware specifics. For this project since it has two water motors I recommend a power source of more than 10 Volts for desired revolutions. For the Arduino I would also recommend one that comes with more digital I/O pins since the limited number of digital pins restricted the possibilities of the project however the Arduino UNO R4 WiFi still works wonders. Below you will see various prototypes that can help you. As you can see with the H-Bridge it has two motors with positive and negative powers, and a central pin for the battery pack that is sending the voltage to each motor, along with the sensor connected to the Arduino power supply as well as the H bridge itself and the flow meter, however make sure the power for the motors and the power for the sensor remains separate.

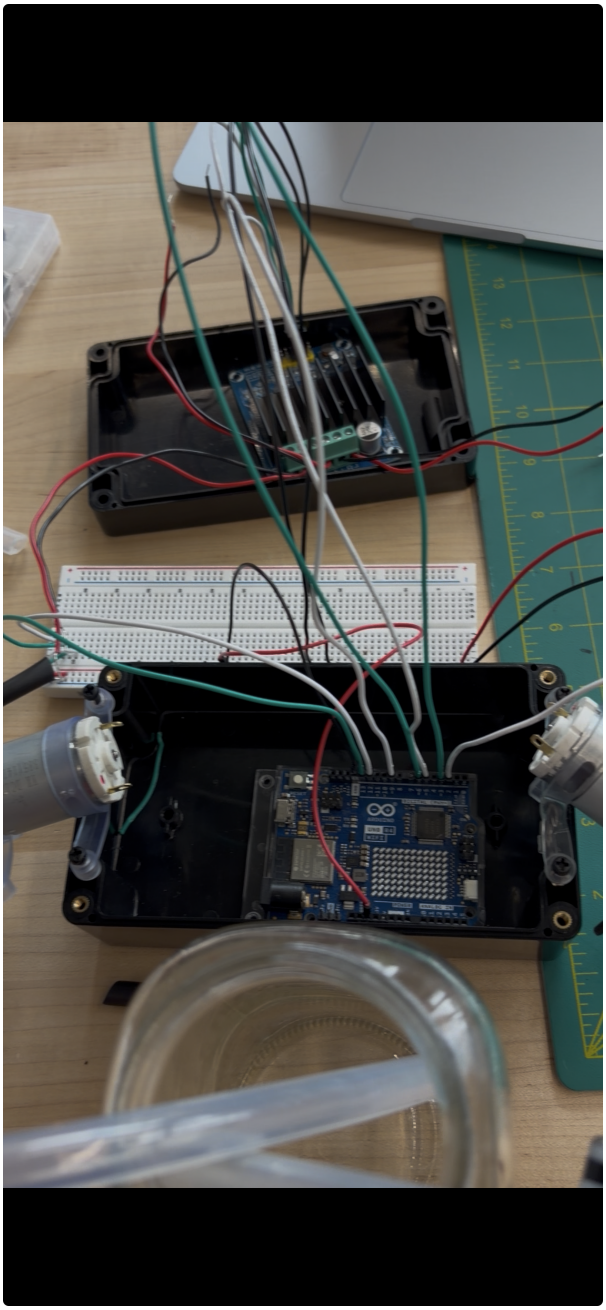

Figure 3.1: Hardware setup view 1

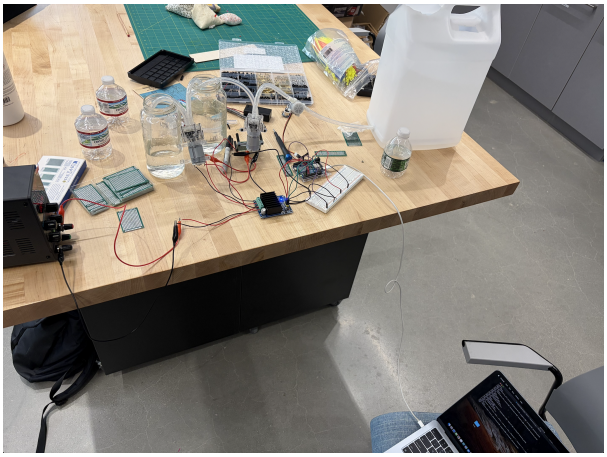

Figure 3.2: Hardware setup view 2

3.1 System Startup

The Hardware should be checked and powered in the correct way, and for the flow meter used make sure it has the arrow pointing in the direction of the flow of water. Once the Arduino is connected to the computer open the Serial monitor in the Ino app to make sure that the Arduino successfully connected to the WiFi using the Secrets file to you can put the name and password of the desired network. Make sure you have the correct IP address, with the Ino file it should be printed out and it should match the one in the Ocaml controller since they need to be matched up for the controller to connect to the Arduino.

3.2 OCaml Startup

Using Dune you open the dispenser scrip via the Command Line Interface and run the command below with your own IP dress and port:

```
dune exec ./dispenser.exe -- --host 192.168.1.142 --port 4080
```

After executing this command an interactive command prompt, which shows use a plethora of commands that can be used without editing the Arduino code:

User Command	**Purpose**	**Arduino Message**
`hello`	Checks communication	`hello`
`status`	Reads current system status	`status`
`measure`	Reads ultrasonic distance	`measure`
`stop`	Stops all pump activity	`stop`
`transfer A B 1.5`	Moves 1.5 L from Tank A to Tank B	`transfer,A,B,1.500`
`transfer B A 1.5`	Moves 1.5 L from Tank B to Tank A	`transfer,B,A,1.500`
`pump1 1.5`	Runs Pump 1 for 1.5 L	`transfer,A,B,1.500`
`pump2 1.5`	Runs Pump 2 for 1.5 L	`transfer,B,A,1.500`

Table 3.1: User commands for the OCaml controller

3.3 Shutdown Procedure

To shut down the system, first send the `stop` command to make sure no pump is running. After the Arduino confirms that the pumps have stopped, disconnect the pump power supply. Finally, disconnect the Arduino from the computer or power source.

The system should not be unplugged while a pump is actively transferring water unless it is an emergency. Stopping the pump through software first keeps the transfer state more predictable and makes testing easier.

Chapter 4

Conclusion

This project was near and dear to me, I hope that it allows people in the Clark community to become more expressive and creative with whatever project they want to do. I want to extend the biggest thank you to Professor Holness for overseeing this project's entire development from start to finish and I hope that more students are able to fully realize their own ambitions. The beauty of this project is the fact that you can cater it to your specific needs without compromising the entire system, i quick change here and there and you can have a very different project with its own set of values and rewards. For the future I hope this project can become into an agriculture standpoint for Clark and use these ideas to build gardens, greenhouses, and any system that can have a positive impact on the Clark environment.