MES: Plant Controller Report - DRAFT

Friedemann Drepper

October 13, 2022

Contents

1 Application Description

The watering system is used to water plants at home. When active, it checks the soil moisture around the plant. If the moisture level is below a preset value it will water the plant until the soil is moist again. When inactive for a while, it will go to sleep. Every 10 minutes it wakes up and checks for the soil moisture, the water level of the tank and then goes back to sleep again. If the soil moisture is below the set value, it will water the plant and go into active mode. If the tank is empty, the system will notify the user by blinking an LED and displaying a warning message on the LCD screen while also preventing the system from working until the tank is filled again.

The user has two ways to interact with the system. First, by pressing a button the system gets active (when in sleep mode) and displays information on an LCD screen which you can cycle through by pressing the button again. Second, by interfacing to it with a UART Terminal, that is used for debugging and setting the moisture level at which the system should water the plant. It will also activate the system when it is in sleep mode.

For the watering itself, the system uses a small tank and a peristaltic pump. The water level in the tank is monitored by an ultrasonic distance sensor, meaning the system is dependent on knowing the tank size. If the tank is empty, the system will notify the user by blinking an LED and displaying a warning message on the LCD screen.

The system also makes predictions about when the next watering is likely to occur, based on previous times, how much water is needed on average and how long the remaining water in the tank will last. This information is also available on the LCD Screen by cycling through it with the button or by requesting them over the UART Terminal. It can be used to calculate how much water the plant will probably need if you go on vacation.

To find out what soil moisture level is correct you need to calibrate the system with experimental data of dry and moist soil and then figure out a good moisture value.

The full state machine of the system can be found here

2 Hardware Description

The system consists of the Processor, a STM32F446RE connected to the LCD screen, the sensors and the peristaltic pump. The pump is pumping water from the tank to the plant's pot, filled with soil. The tank has the distance sensor mounted on top which measures the distance from sensor to the water surface by sending out ultrasonic waves and registering their reflection from the water. Inside the plant's soil in the pot is the moisture sensor. The Processor also can be connected to a PC to interface with it over a serial terminal console. Lastly, there is a user button and a reset button connected to the processor. The Hardware Block Diagram:

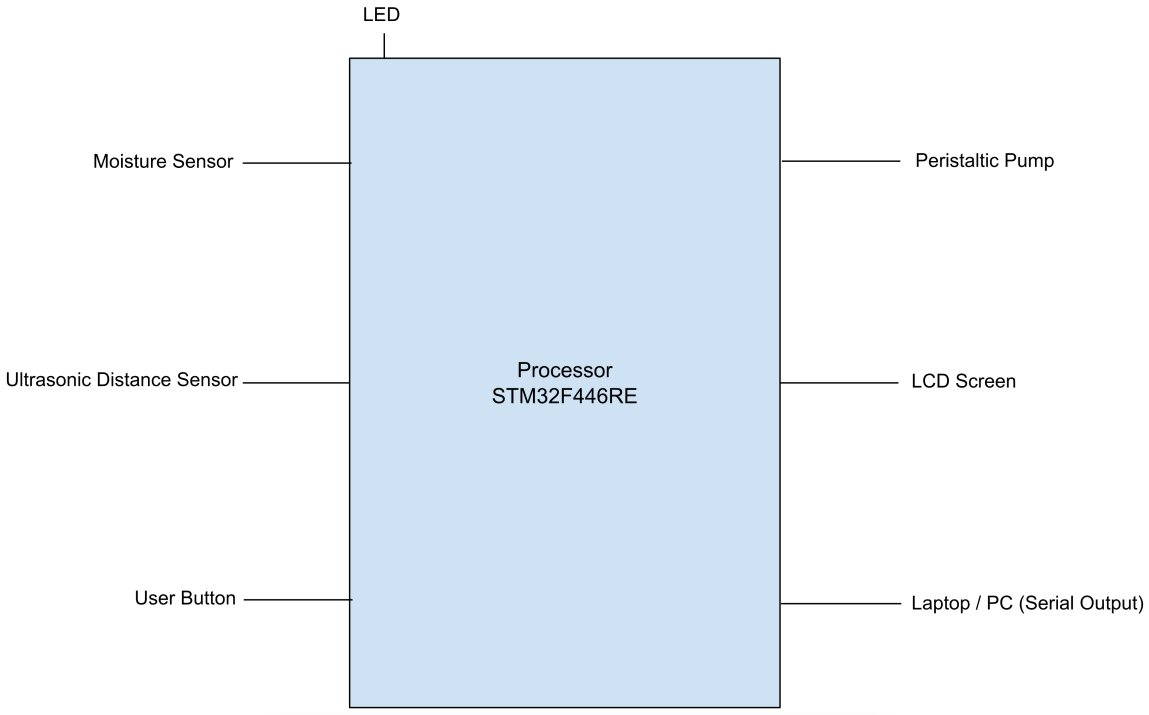

Figure 1: Hardware Block Diagram

A list of still missing hardware / setup:

- Moisture Sensor - Capacitative or Resistive
- Peristaltic Pump
- Water Tank
- Test Plant Pot with Soil
- Ultrasonic Distance Sensor (HC-SR04)

3 Software Description

Starting on a high level, the systems behaviour is documented in its state machine.

States (Rows) / Events (Columns)	No Events happening	RTC wakeup	Idle Timeout	Button Pressed	UART Receives Contents	Tank Empty	Soil is dry
STARTUP	ACTIVE						
ACTIVE	MAKE PREDICTIONS	ACTIVE	ENABLE RTC	CYCLE LCD	PROCESS MESSAGE	ERROR TANK EMPTY	WATERING
ENABLE RTC	SLEEP	SLEEP	SLEEP	SLEEP	SLEEP	SLEEP	SLEEP
SLEEP	SLEEP	PERIODIC CHECK		ACTIVE	ACTIVE	SLEEP	SLEEP
PERIODIC CHECK	SLEEP	PERIODIC CHECK	SLEEP	ACTIVE	ACTIVE	ERROR TANK EMPTY	WATERING
WATERING	ACTIVE	WATERING	WATERING	WATERING	WATERING	ERROR TANK EMPTY	WATERING
CYCLE LCD	ACTIVE	ACTIVE	ACTIVE	ACTIVE	ACTIVE	ACTIVE	ACTIVE
MAKE PREDICTIONS	ACTIVE	ACTIVE	ACTIVE	ACTIVE	ACTIVE	ACTIVE	ACTIVE
PROCESS MESSAGE	ACTIVE	ACTIVE	ACTIVE	ACTIVE	ACTIVE	ACTIVE	ACTIVE
ERROR TANK EMPTY	ACTIVE	ERROR TANK EMPTY	ERROR TANK EMPTY	ERROR TANK EMPTY	ERROR TANK EMPTY	ERROR TANK EMPTY	ERROR TANK EMPTY

Figure 2: System State Machine

It is an event-centric state machine where the events are the user interactions and the sensor readings. This means that the system always needs to keep track of the sensors and compare them to the set values for tank emptiness and soil dryness. The system's state machine also is implemented in it to keep track of what mode it is in right now. For a visual overview see the Software Block Diagram.

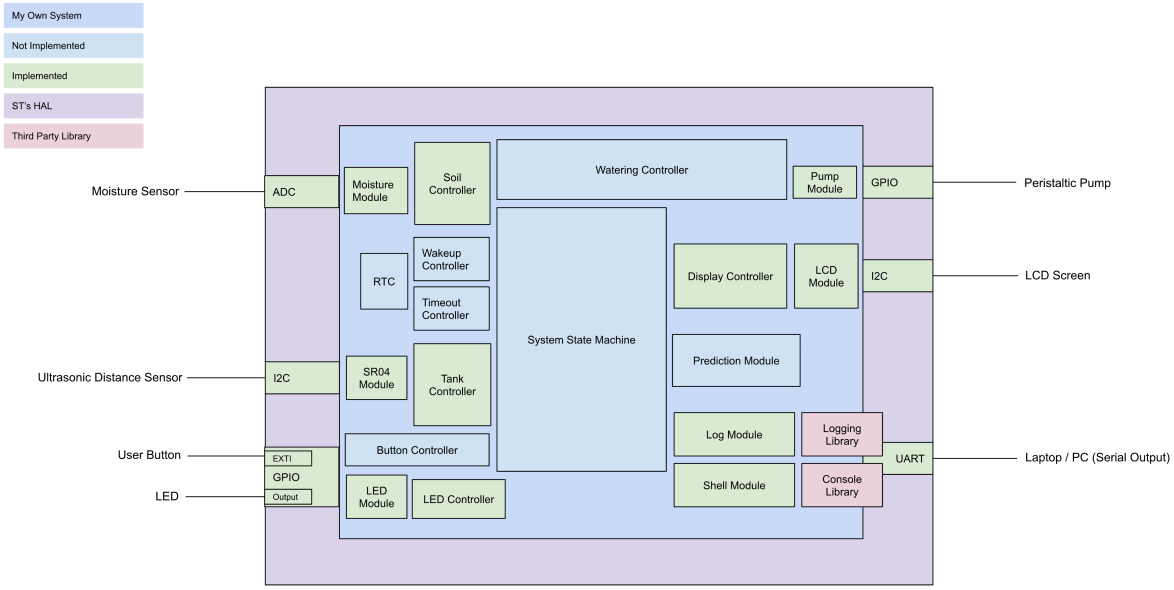

Figure 3: Software Block Diagram

I generated all the code regarding system setup, interrupt handlers and peripherals with the CubeMX tool by ST. I also use the CubeHAL and LL drivers to program the hardware, though I created subsystems that work as adapters to be platform independent, allowing easier migration to another processor by ST (or even another vendor). Talking of vendors, inside the `vendor` folder are all third party libraries found I used. I use the console and logging library from anchor and (hopefully) the lwrb library. Ceedling is the testing framework I use for Unit Tests by ThrowTheSwitch. The only two files I changed inside the `Core` folder, which is the generated code, are `main.c` which contains the main loop and `stm32f4xx_it.c` which contains the interrupt. All files in `inc` and `src` are my code, containing sensor drivers, subsystems, modules and controllers. The Unit Tests are found in `test`.

This being the draft for the final project report, I can't actually describe how the drivers, modules, controllers etc. work yet. Instead I will use this chapter to make a list of the tasks I want to get done:

- LED that blinks slowly to show the processor is not broken and is flashed with firmware
- Subsystems for all peripherals of the processor that I use: UART, Timers, ADC, SPI and GPIO
- Modules for LED, LCD screen, Moisture Sensor, Distance Sensor and Water pump that provide object structs and methods for these structs
- Controllers to use the modules and keep track of moisture level and water level and cycling the information displayed on LCD screen
- Scheduler which is used to periodically query the sensors when the system is in active mode
- A console and a logging system
- A module that keeps track of sensor data of the last times the plant got watered to make predictions about the future
- Implement the system state machine:
  - sleep mode and RTC wake up to check sensor data
  - watering controller that takes care of watering the plant
  - tank empty error mode
  - an idle timeout in active mode

4 Build Instructions

I am using the CubeMX Tool by ST to generate code setting up peripherals, startup code, linker scripts etc. For compiling and debugging I use the `arm-none-eabi` toolchain (`gcc` and `gdb`). All my code is written in C, so there is no need for any C++ compiler. My CMakeLists is customized to support a project file structure that separates generated code and third-party libraries from self written code. For further detail about the file structure, see the Software Description section. As operating system for my build platfrom I use MacOS and as microprocessor I use the STM32F446RE mounted onto the corresponding Nucleo Board (Nucleo-F446RE).

For Unit Tests, I use the Ceedling framework by ThrowTheSwitch.

5 Future

Some Ideas what could be done to enhance the project but I won't have time to do

- Using the F-Series by ST is an overkill for such a simple system, especially regarding power consumption. Therefore using the L-Series of processors by ST which are designed for low power applications would make a lot of sense. For ease of completing the project I decided to go with a board I already had. Because of this, I am already trying to make the main system logic as independent of the underlying chip as possible.
- In future the it'd also make sense if the system would be battery powered. For plants on a balcony or in a garden, adding a solar cell to charge the battery would make sense.
- The system could send its information wireless to a central system like a raspberry pi, which makes the information online available to monitor the plants remotely. Even adding a custom watering command would be possible.
- As a hobby project the form of prototyping it with development boards and breadboards is a good and easy way, though if the system should be commercially available in the future all the circuitry should go on one PCB (having the moisture sensor attached). Also the tank, pumps and tubes should be sold with it, making it easier to use and give room for making it compact and easy to set up.

6 Self Assessment

This Section will be left empty, as there is nothing to put in here for the draft.