

TTK8 Project: Automatic Watering System

Anders Kristoffersen

November 17, 2025

Contents

| | | |
|----------|--|----------|
| 1 | List of Abbreviations | 2 |
| 2 | Introduction | 2 |
| 3 | Specification (Demands and Acceptance Criteria) | 3 |
| 4 | Design of Automatic Watering System | 4 |
| 4.1 | System Architecture (Top-Down) | 4 |
| 4.2 | Electrical Connections | 5 |
| 4.3 | Code Excerpts | 5 |
| 5 | Implementation Details | 6 |
| 5.1 | Source Files and Responsibilities | 6 |
| 5.2 | Threshold Watering Logic | 6 |
| 5.3 | UART Test Interface (Test-Only) | 6 |
| 5.4 | Key Algorithms and Flow | 6 |
| 6 | Testing of Automatic Watering System | 7 |
| 6.1 | Test Plan and Acceptance Tests | 7 |
| 6.2 | Manual Test Procedure | 7 |
| 6.3 | Test Results | 7 |
| 7 | Conclusion & Future Work | 7 |

1 List of Abbreviations

- **ADC**: Analog-to-Digital Converter
- **MCU**: Microcontroller Unit
- **GPIO**: General Purpose Input/Output
- **UART**: Universal Asynchronous Receiver/Transmitter
- **PSU**: Power Supply Unit
- **GND**: Ground
- **VCC**: Supply Voltage
- **RTC**: Real Time Counter
- **PIT**: Periodic Interrupt Timer
- **LCD**: Liquid Crystal Display

2 Introduction

This document describes the automatic watering system developed in this course. It explains the system from top level design down to implementation details in the firmware, lists the functional specification and acceptance criteria, documents the design (hardware and firmware), and outlines the testing approach and results. The firmware runs on a **ATmega4809** MCU and controls a **WHADDA Water Pump** via a relay driver. UART communication is present strictly for test/debug interaction (manual relay control and moisture reads) and is not used during normal autonomous operation. Firmware is reflected through selected code excerpts, but complete code can be found at <https://github.com/AndersKristof/ttk8>.

Figures

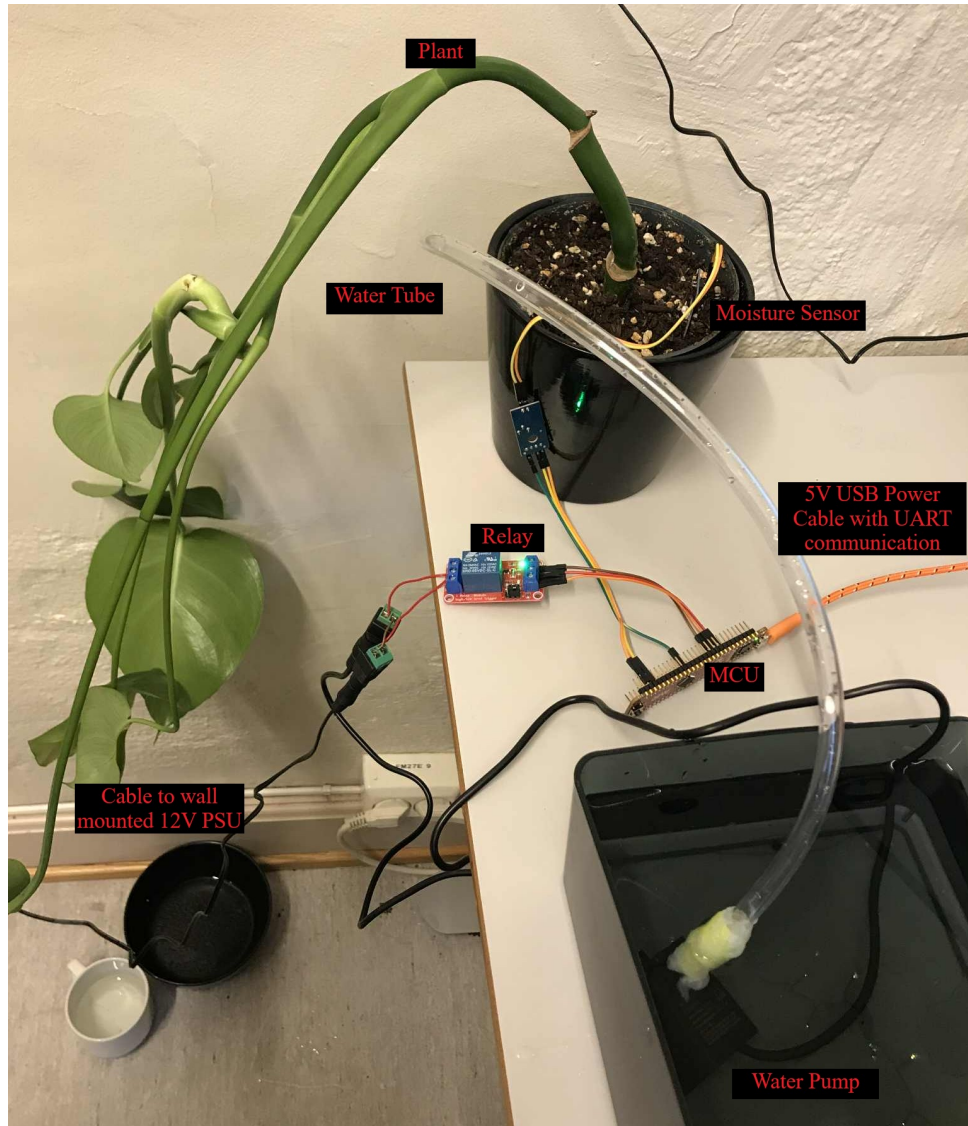


Figure 1: System setup.

3 Specification (Demands and Acceptance Criteria)

Purpose

Automatically water plants by measuring soil moisture and energizing the pump (via relay) for a fixed duration when moisture falls below a configured percentage threshold.

Acceptance Criteria

- In autonomous mode (`TEST_MODE_ACTIVE == 0`), sample moisture once per `SAMPLE_INTERVAL_MS` and water for `WATER_DURATION_MS` if below threshold.
- In test mode (`TEST_MODE_ACTIVE == 1`), UART commands operate the system without unintended relay activation.
- Relay remains OFF at power-up until a valid watering condition or explicit test command.

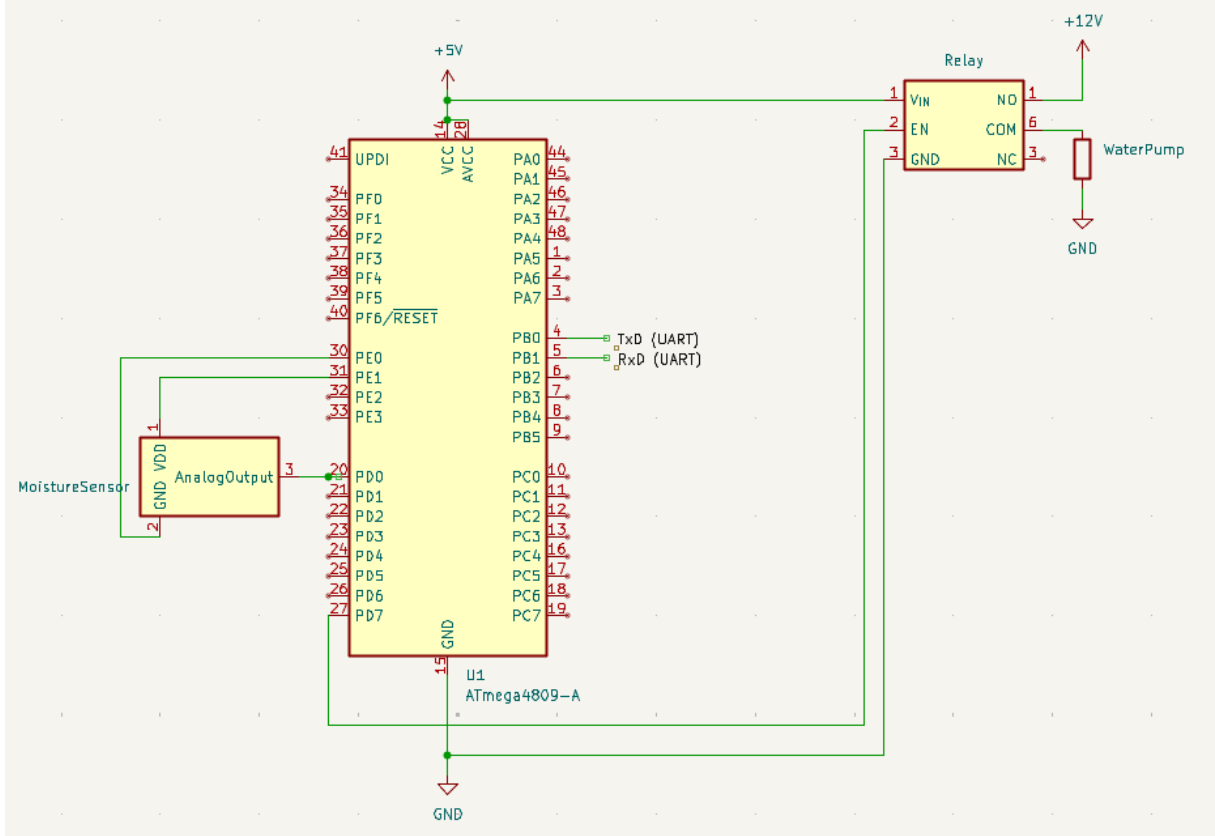


Figure 2: Electrical connections schematic.

- Moisture percent computation is monotonic with sensor changes.

Current Implemented Demands

- Read soil moisture via 8-bit ADC and convert raw value to a percentage.
- Drive relay controlling the WHADDA Water Pump for `WATER_DURATION_MS` when moisture level is below `MOISTURE_THRESHOLD_PERCENT`.
- Provide a UART test interface with commands for reading out current moisture level ('c'), toggling the relay controlling the pump ('t'), and activating the pump for the specified watering interval ('r').
- Relay defaults to OFF on startup and after watering delay.

4 Design of Automatic Watering System

4.1 System Architecture (Top-Down)

- **Sensor Layer:** Soil moisture probe powered by dedicated GPIO pins (PE0 and PE1) with analog output routed to internal ADC0 (AIN0 = PD0).
- **Control Layer:** ATmega4809 executes threshold-based loop; converts raw ADC value to percent, compares with `MOISTURE_THRESHOLD_PERCENT`.
- **Actuation Layer:** Relay connected to PD7 drives the WHADDA Water Pump which has its own 12V PSU.
- **Monitoring & Test Interface:** UART (USART3) enabled only when `TEST_MODE_ACTIVE==1`.

4.2 Electrical Connections

- PD0 (AIN0) → Sensor analog input
- PE0 / PE1 → Sensor power output
- PD7 → Relay driver output
- PB0 (TX), PB1 (RX) → UART test interface (test mode only)
- Shared 5V VCC and GND between MCU, relay controls and moisture sensor
- Own VCC and GND for Water Pump from 12V PSU

4.3 Code Excerpts

Complete firmware code can be found at the project repository; <https://github.com/AndersKristof/ttk8>.

Main Autonomous Loop

```
1 while (1) {
2     moisture_level_in_percent = 100 * abs(adc_get_result() - 255) / 255;
3     if (moisture_level_in_percent < MOISTURE_THRESHOLD_PERCENT && !
4         TEST_MODE_ACTIVE) {
5         relay_on();
6         _delay_ms(WATER_DURATION_MS);
7         relay_off();
8     }
9     _delay_ms(SAMPLE_INTERVAL_MS);
10 }
```

UART Command ISR (Test Mode Only)

```
1 ISR(USART3_RXC_vect) {
2     uint8_t byte = uart_receive_byte();
3     if (byte == 't') { relay_toggle(); }
4     else if (byte == 'r') { relay_on(); _delay_ms(WATER_DURATION_MS);
5         relay_off(); }
6     else if (byte == 'c') {
7         moisture_level_in_percent = 100 * abs(adc_get_result() - 255) /
8         255;
9         // print moisture
10    }
11    // unknown commands are reported back in test mode
12 }
```

Relay Control

```
1 void relay_init() { PORTD_DIRSET = PIN7_bm; PORTD_OUTCLR = PIN7_bm; }
2 void relay_on()   { PORTD_OUTSET = PIN7_bm; }
3 void relay_off()  { PORTD_OUTCLR = PIN7_bm; }
4 void relay_toggle() { PORTD_OUTTGL = PIN7_bm; }
```

ADC Sampling

```
1 uint8_t adc_get_result() {
2     ADC0_COMMAND = ADC_STCONV_bm;
3     while (!(ADC0_INTFLAGS & ADC_RESRDY_bm));
4     return ADC0_RES; // 8-bit value
5 }
```

Moisture Sensor Power Setup

```
1 void moisture_sensor_init() {  
2     PORTE_DIRSET = (PIN0_bm | PIN1_bm);  
3     PORTE_OUTSET = PIN1_bm; // V+  
4     PORTE_OUTCLR = PIN0_bm; // GND/ref  
5 }
```

5 Implementation Details

5.1 Source Files and Responsibilities

- **main.c**: Entry point; initializes modules and runs threshold loop; enables UART and interrupts only in test mode.
- **config.h**: Central configuration macros (threshold, durations, interval, test flag, UART baud calc).
- **moisture_sensor.c/.h**: GPIO setup to power sensor.
- **adc.c/.h**: ADC0 init and blocking conversion; percent computed externally.
- **relay.c/.h**: PD7 relay driver (init/on/off/toggle).
- **uart.c/.h**: USART3 driver for test mode commands and output.
- **clkctrl.c/.h**: Clock source configuration (20 MHz internal oscillator with prescaler, running at 2 MHz).

5.2 Threshold Watering Logic

Single moisture percent threshold (**MOISTURE_THRESHOLD_PERCENT**). Below threshold: relay ON for **WATER_DURATION_MS** then OFF; loop sleeps via blocking delay.

5.3 UART Test Interface (Test-Only)

Active only when **TEST_MODE_ACTIVE==1**:

- **t**: Toggle relay
- **r**: One watering sequence
- **c**: Read and print moisture percent
- **Unknown**: Error message

Autonomous mode omits UART to reduce power and noise.

5.4 Key Algorithms and Flow

- **Sampling**: Single 8-bit ADC conversion per loop.
- **Percent Conversion**: $\text{percent} = 100 * \text{abs}(\text{adc} - 255) / 255$ (inverted scale assumption).
- **Watering**: Fixed ON duration.
- **Safety**: Relay OFF in **relay_init()**.

6 Testing of Automatic Watering System

6.1 Test Plan and Acceptance Tests

- UART commands: In test mode, verify `t`, `r`, `c` behaviors.
- Sensor check: Check sensor output for different moisture-levels; find appropriate threshold level, `MOISTURE_THRESHOLD_PERCENT`.
- Flow rate: Measure volume amount of water per second of ON time for appropriate `WATER_DURATION_MS` value.
- Relay safety: Power cycle; relay remains OFF until condition/command.
- Threshold activation: Force dry condition; confirm relay activation and ON duration matches `WATER_DURATION_MS`.

6.2 Manual Test Procedure

1. Set `TEST_MODE_ACTIVE=1`; build/flash.
2. Connect serial terminal at configured baud.
3. Issue `c` while varying moisture; record readings.
4. Issue `r`; measure ON duration. Measure water volume pumped to choose appropriate ON time.
5. Issue `t` several times; confirm stable toggling.
6. Set `TEST_MODE_ACTIVE=0`; reflash; simulate dry condition by pulling sensor out in the air and observe autonomous watering.

6.3 Test Results

- UART commands: Commands behave as expected.
- Sensor check: Observe low (0-2%) output when sensor is out of soil, high (97%) when sensor poles are directly connected, and medium (5-60%) when sensor is placed in soil. Soil appears ready for watering at around 15% sensor output, choose this as `MOISTURE_THRESHOLD_PERCENT`.
- Flow rate: Measured 0.2L of water per 1 seconds of ON time. 0.1L of water is an appropriate amount for each activation so choose `WATER_DURATION_MS` to be 500.
- Relay safety: Relay is not activated during power cycle.
- Threshold activation: As expected. Watering is activated every `SAMPLE_INTERVAL_MS` (60s) when sensor is detecting dry conditions.

7 Conclusion & Future Work

Current system delivers a threshold-based watering cycle with modular firmware structure (`main`, `config`, `moisture_sensor`, `adc`, `relay`, `uart`, `clkctrl`).

Planned Enhancements

- Running in low-power mode using the sleep controller peripheral with RTC/PIT-based wake-up replacing blocking delays.
- Runtime UART configuration commands (threshold/duration adjustments).

- LCD screen for live moisture-level tracking.
- Purpose-built electronics container.
- Purpose-built water container.
- Complete powering of the system from wall-mounted PSU, with MCU, relay and sensor powered through a voltage regulator.