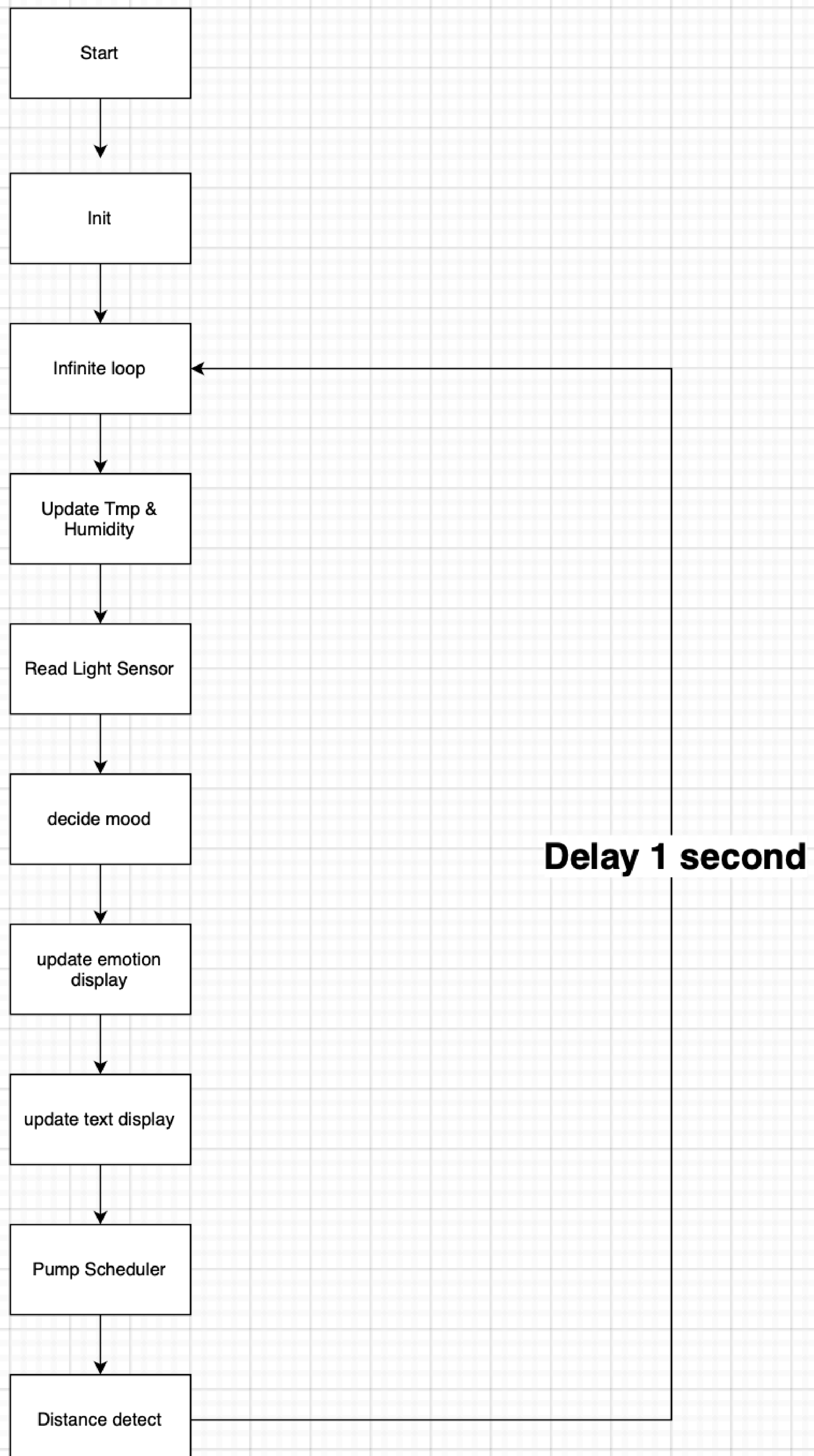
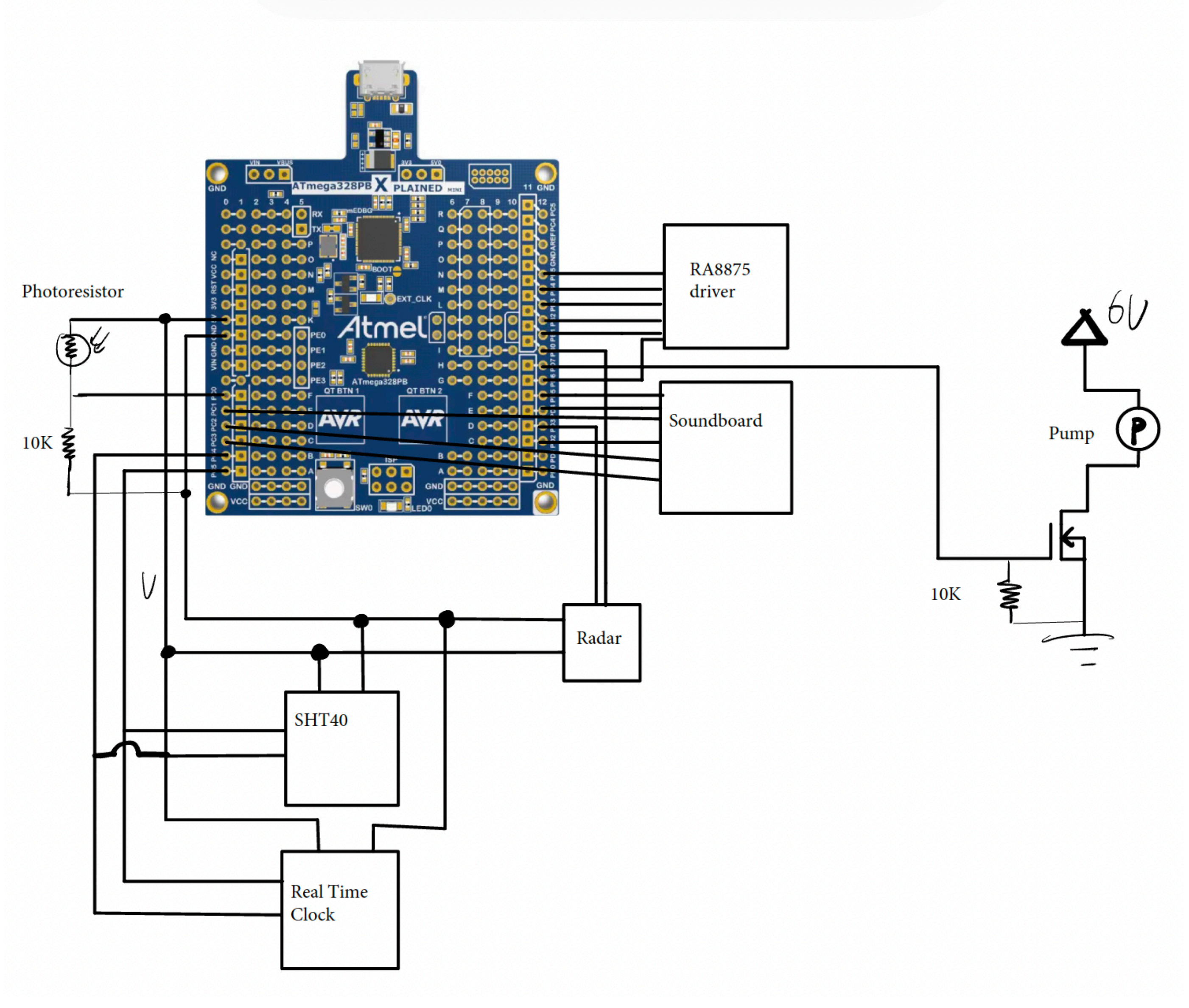
# MVP Demo Report - Team 3 Moodo

## 1. Show a system block diagram & explain the hardware implementation.


Start
Init
Infinite loop
Update Tmp & Humidity
Read Light Sensor
decide mood
update emotion display
update text display
Pump Scheduler
Distance detect
Delay 1 second

Below is our hardware connection diagram. For simplicity, the RA8875 driver and the LCD module are shown as an integrated unit using a single 40-pin TFT connector. The soundboard and speaker are connected via their signal line, 5V power, and ground.



## 2. Explain your firmware implementation, including application logic and critical drivers you've written.

Our firmware is organized as a simple loop running at ~1 Hz, with all time-critical sensing handled by hardware peripherals and interrupts.

The init() routine configures the core drivers (UART, I2C/TWI, SPI LCD, ADC, timers, and GPIOs), and the while(1) loop periodically polls sensors, updates sensor reading variables (light, distance, humidity), triggers sounds, changes LCD display and runs the pump scheduler. Based on the sensor readings, the system determines Moodo's mood. Emotion display updates only when mood changes, avoiding unnecessary redraws."

After initialization, we play a power-up greeting (SOUND_HELLO) and then enter the main loop, which performs one full sensing/logic cycle per second (_delay_ms(1000)). Each loop (~1 s) performs the following:

1. Update environment (temperature, humidity, light).
2. Light logic: detect LIGHT_ON / LIGHT_OFF transitions and trigger sleep sound after prolonged darkness.
3. Humidity logic: from non-dry → dry, play "THIRSTY."
4. Distance logic: near → greet the user; prolonged near → "STAND UP;" far after interaction → "SLEEP."
5. Mood computation: decide_mood() maps environmental conditions to one of six moods (sleep, happy, hot, cold, thirsty, normal).
6. LCD update: refreshes emotion graphics and displays time + sensor values.
7. Pump scheduling: simple time-based cycle toggling PD7 to water the plant periodically

Driver files: ds1307: Real-time clock driver. Handles issuing periodic I²C commands and reading the current time from the DS1307 RTC module.

ra8875_drv.h / ra8875_gfx: Low-level RA8875 display controller driver and higher-level graphics utilities. These files provide primitives such as drawing pixels, lines, text, and handling screen updates for the TFT LCD.

twi0.h: Implements the I²C (TWI0) interface used for communication with devices such as the SHT40 sensor and DS1307 RTC.

uart.h: UART serial driver. Not required for final functionality, but included during development to enable debugging via printf and serial logging.

sht40.h: Driver for the SHT40 temperature and humidity sensor. Integrates command transmission, data acquisition, and conversion of raw sensor readings into physical values.

## 3. Video Demo.

Overall_demo

Pump_demo

Sound_demo

## 4. Have you achieved some or all of your Software Requirements Specification (SRS)?

**Functionality**

| ID | Description | MVP demo result |
|---|---|---|
| SRS-01 | On capacitive touch, play the corresponding voice prompt and switch the on-screen expression (≥95/100 touches correctly recognized; audio start latency ≤300 ms). | **Failed.** Adding a capacitive touchscreen exceeded our hardware budget, so this feature was not implemented |
| SRS-02 | If a user remains within ≤60 cm for ≥1 hour, play a reminder (timer-based; cadence error ≤±5s). | **Achieved.** The RTC has been validated with a 1 s/hour cadence error which meet our requirement, and the system successfully triggers a voice reminder when the user stays nearby for an extended period. |
| SRS-03 | The LCD provides a face page and a data page; data page shows temperature, humidity, and time (page switch + data refresh period ≤2 s). | **Achieved.** Temperature and humidity readings update accurately, and the data refresh cycle is now 1 second. |
| SRS-04 | When lux falls below the “night” threshold, enter Night Mode and play a “good-night” prompt (configurable threshold; state-change hysteresis ≤2 s; false triggers ≤1/hour). | **Achieved.** The system now reliably detects light-intensity changes and transitions into Night Mode as expected. |
| SRS-05 | Via ESP8266 HTTP, the phone can view T/H data (LAN end-to-end | **Failed. Wireless connectivity was not implemented because all** |

| ID | Description | MVP demo result |
|---|---|---|
| | latency ≤3 s; packet loss ≤1% per 5 min). | **available pins are already allocated for other essential modules, so the system currently operates locally only.** |

## 5. Have you achieved some or all of your Hardware Requirements Specification (HRS)?

| ID | Description | MVP demo result |
|---|---|---|
| HRS-01 | Use ATmega328PB Xplained Mini @16 MHz with at least 1×TWI, 1×UART, ADC, and a 16-bit timer/input-capture. | Achieved |
| HRS-03 | One TRIG and one ECHO routed to MCU digital I/O and timer input-capture (20–200 cm average error ≤±2 cm). | Achieved |
| HRS-04 | Mini MP3 module + microSD, UART-controlled, feeding a small power amp and speaker (≥70 dB SPL at 1 m; UART BER ≤$10^{-5}$). | Achieved. (no micro SD need, soundboard has a 16MB flash) |
| HRS-05 | 0.96″ OLED (I²C or SPI) for face/data pages; status LEDs for network/fault (readable from 0–40 °C). | Achieved. We upgrade to a 5.0" LCD display |
| HRS-06 | ESP8266 over UART providing HTTP service. | Failed |
| HRS-07 | Enclosure/planter integrates cable channels; sensor wiring are routed and strain-relieved appropriately. | Not completed yet. |

## 6. Show how you collected data and the outcomes.

**Real-Time Clock (RTC):** To verify the accuracy of the RTC, we used a smartphone clock as the reference time source. Across three separate one-hour tests, the RTC's drift remained within ±1 second per hour, which meets our timing precision requirement.

**Light Intensity Sensor:** We evaluated the light sensor by manually blocking it with a finger and observing its response to sudden changes in illumination—it reacted instantly. We also turned off all room light sources to simulate nighttime conditions. Under these conditions, the analog reading consistently dropped below 300, so we set 300 as the threshold between "day mode" and "night mode." This threshold is stable and repeatable across our tests.

**Temperature and Humidity Sensor:** Although we lack laboratory-grade equipment for absolute accuracy validation, we verified the sensor qualitatively. The readings remained stable and reasonable during normal operation. When a finger was brought close to the sensor, both temperature and humidity increased gradually, as expected. However, we currently cannot quantify the measurement error or test cold-side accuracy.

**Ultrasonic (Distance Sensor):** We tested the ultrasonic sensor at multiple distance ranges: 10–50 cm, 50–100 cm, and 100–200 cm.

1. At 10–50 cm, performance was excellent with errors within ±3 cm.
2. At 50–100 cm, errors increased to 5–15 cm.
3. At 100–200 cm, the error margin grew to 20–30 cm. These results match the typical behavior of low-cost ultrasonic modules, where accuracy decreases with distance

## 7. Show off the remaining elements that will make your project whole: mechanical casework, supporting graphical user interface (GUI), web portal, etc.

**Mechanical Casework:** We will design a protective enclosure that prevents any circuits from being exposed to the user. The case will also improve the overall reliability by reinforcing the physical connections between the ATmega328PB and all attached modules.

**Connection & Structural Improvements:** Our goal is to move away from breadboards and loose jumper wires, replacing them with more secure and permanent wiring to ensure stable long-term operation.

**Power Solution:** Currently, the ATmega328PB and water pump are powered through a laptop and an external power supply. For the final build, we plan to use a power bank to simulate a self-contained, battery-powered system, allowing our project to operate safely, portably, and independently without external equipment.

## 8. What is the riskiest part remaining of your project?

Our enclosure must be waterproof, because unlike many other projects, our system includes an active watering mechanism. Even a single drop of water contacting the exposed circuit or metal connections could cause short-circuiting and permanently damage the electronics.

## 9. How do you plan to de-risk this?

Therefore, it is essential for us to design a safe, sealed, and reliable protective enclosure that prevents water while still maintaining a clean, simple, and appealing appearance. In addition, we plan to eliminate the use of breadboards and jumper wires, as they are prone to loose connections and are not suitable for long-term stability. Instead, we will transition to more secure and permanent wiring solutions to ensure robustness and safety in the final system.

## 10. What questions or help do you need from the teaching team?

What are some recommended approaches or best practices for building a stable, safe, and portable power system for our ATmega328PB and pump?