**Automated Plant Watering System**

**(APWS)**

**Purpose:**

APWS (Automated Plant Watering System) is designed, first and foremost, to remove the need of human intervention in the process of watering a collection of plants. (chosen arbitrary) More importantly, its utility comes from the fact that we, as humans, cannot really guess/ predict (with rare exceptions), exactly when a certain plant “starts to need water”. This expression translates into: “the soil moisture has dropped below acceptable levels; if the soil is not watered, then the plant will start to slowly die”.

By the time we finally water the plant, regardless if it survived or not, the point is that we, as humans, took a guess: “the plant needs water”. We did not know INSTANTLY when the plant’s life is endangered because of the low soil moisture.

Our project comes as a solution to this problem. Swiftly explained:

* the plants, placed in a matrix formation (not necessarily full rows), have **soil moisture sensors** placed in (one for each)
* the moment a certain plant’s moisture plummets, the data is transmitted to a “**data harvesting point**”; (there is more than one point) there, the said data will be sent to a “Control Tower” (reffered from now on as CT), situated outside the matrix-shaped pot formation
* from there, a robot (car) will receive data from the CT, and **will go to the endangered plant’s location**, and it will water it. How?
* well, using a **rotary hose** (why rotary, we’ll see in a moment), connected to a **water container**, that the robot will drag after him
* when the mobile container is **empty,** the robot will head to the position of a larger, **fixed container**; the mobile one is **refilled**, and the robot can continue its activity.

**Schematic:**

Diagram

Description automatically generated

**Observations:**

* the components of the system are placed as in the picture above
* each **data harvesting point** mentioned earlier is an Arduino Nano
* they are **located between columns**, one for each two columns; the reason is: because of the rotary hose, the robot can water a plant to either its left or right; meaning, if the robot moves along the right side of a column, it has no reason to move along ts left side. So we use this **empty space to place the data harvesting points**. Why?
* if we would connect the sensors DIRECTLY to the control tower, we would need lots of wires that would go under the platform that supports the pots, the robot, and the mobile water container; all would go in the direction of the CT, obviously; lots of wires (that are not thouroughly isolated) will form an **electric field that will affect the transmitted data**
* instead, a wire will be used for each harvesting point; so, a wire for two columns of pots
* the robot will follow the path drawn in yellow, which in reality is tape (black in colour)
* IMPORTANT: the **robot cannot perform a 180o turn**, because of the combined length of the robot and the mobile container
* Robot-stop: color/ distance sensors on car.
* the status of a moisture sensors can be checked in a web interface (the values will be saved in a database)
* in case an obstacle is met, it is treated as an anomaly; this will be signaled to the CT
* Red tape at fixed container.
* in case the mobile water container is empty, a water level sensor will signal to the CT that a refill is needed; so, the robot will move just across the fixed container then, the valve will close, and the robot will return to its normal functionality

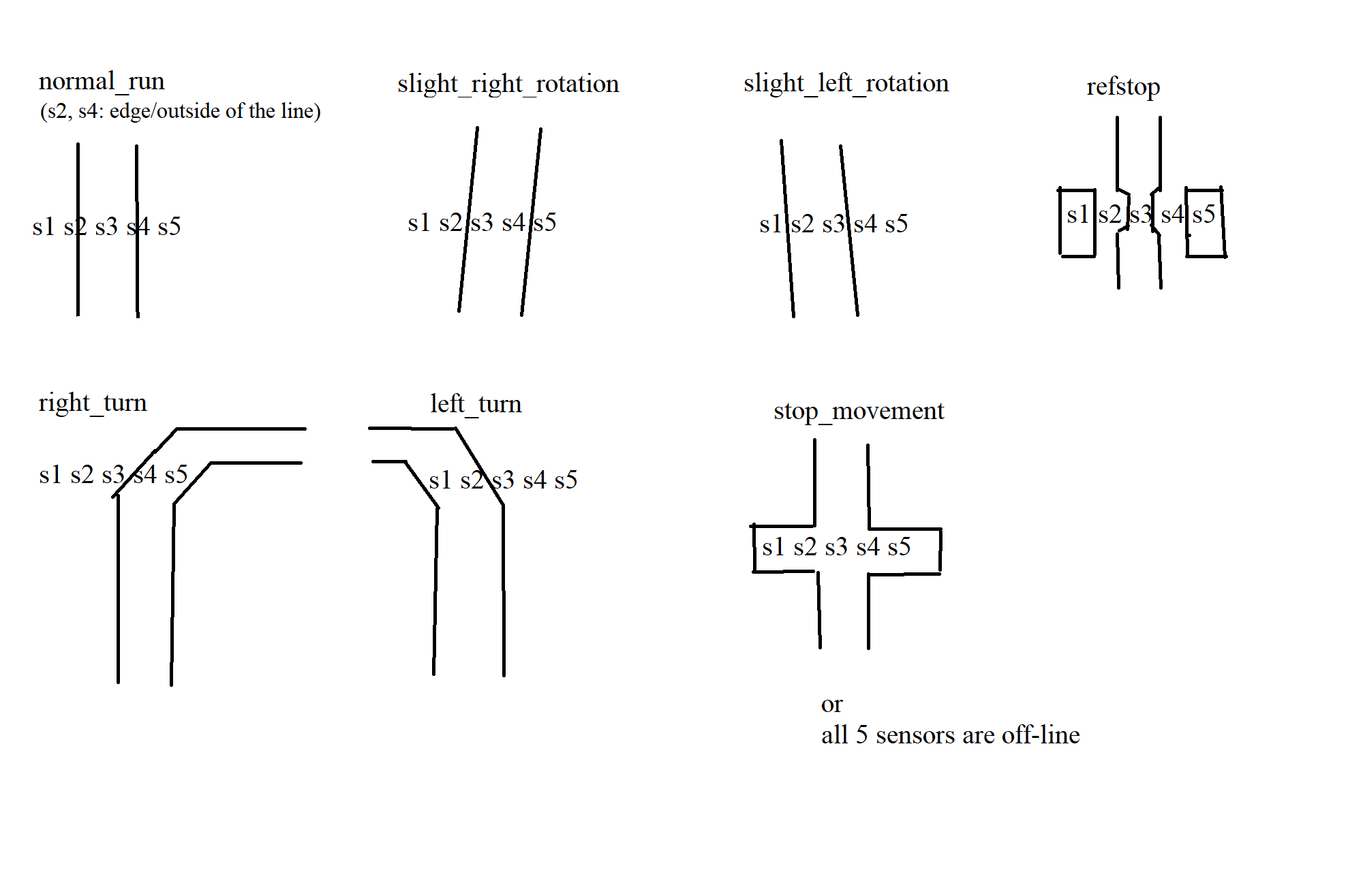
**Workflow, observations (regarding code snippets, algorithms, relations between different components/ Arduino boards, etc):**

1. **Wooden board & car behaviour (line following, plant watering)**
2. **Control Tower & data management (send to Car, receive from DHP, store in Firebase-last one TBA)**
3. **Wooden board & car behaviour (line following, plant watering):**

**Diagram, engineering drawing

Description automatically generated**

The car will start running from the starting point, and alongside the road, it will encounter a series of line formations, which are presented relative to each of the line follower’s 5 sensors (assuming our point of view is the top of the car, s1 is the left-most sensor, s5 the right-most one, etc.), accompanied by a suggestive function name:



The behaviour of the car when encountering each of the situations mentioned above is:

* normal\_run:
  + wheel spin direction: all 4 FORWARD
  + wheel speed: 100 (because apparently, it is the minimum speed at which the motors will spin)
  + result: car moves forward at moderate speed
* slight\_right\_rotation:
  + wheel spin direction: left-side-FORWARD, right-side-BACKWARD
  + wheel speed: 200 (avoid higher speeds; this function is used to correct the car’s position on line)
  + result: car rotates to right relative to its central axis, dragging the trailer by its side, as a result
* slight\_left\_rotation:
  + wheel spin direction: left-side-BACKWARD, right-side-FORWARD
  + wheel speed: 200 (avoid higher speeds; this function is used to correct the car’s position on line)
  + result: car rotates to left relative to its central axis, dragging the trailer by its side, as a result
* right\_turn:
  + wheel spin direction: left-side-FORWARD, right-side-BACKWARD
  + wheel speed: 255
  + result: a quicker spin, to the right
* left\_turn:
  + wheel spin direction: left-side-BACKWARD, right-side-FORWARD
  + wheel speed: 255
  + result: a quicker spin, to the left
* stop\_movement:
  + wheel spin direction: all 4 RELEASEd
  + wheel speed: 0
  + result: the car stops its movement
  + usage: allow the car to halt movement, in order to wet a pot, or if it accidentaly leaves the circuit
* refstop:
  + as in the case of stop\_movement, the car stops in its tracks
  + the function also posesses an additional property, which is used to refill the trailer if it becomes empty; we will detail this aspect later in our presentation.

Regarding how the robot recognises at what pot to stop, we make use of 2 variables potCount and potStops:

* potCount: each time the line follower detects a cross formation (s1-5 are all on the line), potCount increases
* potStops:
  + at the beginning of a reading cycle, the Car recieves from the control tower a string of 16 characters, either 0 or 1 (“0101...001”); the positions occupied by ones represent the pots that need to be watered
  + as a result, the car will iterate through this string, and will stop at the pots that represent indexes of ‘1’ characters; it does this by comparison with potCount
* once the robot reaches a pot:
  + it stops movement
  + the stepper will rotate (MORE TO BE ADDED HERE) based on the current index of the string:
    - first 8 characters => rightmost Data Harvesting Point, and as such, by looking at the schematic, we can tell that regardless of the pot number 0-7, the stepper always rotates to LEFT
    - last 8 characters => leftmost DHP, sofor pots 8-15, the stepper rotates to RIGHT
  + the pump opens, and after 3-4 seconds closes
  + the robot runs to the next pot
* if all necessary pots have been watered, the robot will go back to its initial position, stopping at the REFSTOP point (a boolean, need\_refill will tell the difference between the purpose to refill the mobile container, or simply to stop until the next set of data is received from the Control Tower).

1. **DHP, Control Tower & data management (send to Car, receive from DHP, store in Firebase-last one TBA)**

In this section we will explain how exactly the data is collected from the sensors, sent to the Control Tower, and the information compiled and sent to the car (in the previous section we partially mentioned the receiving end of CT->Car relationship).

**Necessary components (aprox.):**

* Ethernet Shield W5100
* 4 x Modul senzor Ultrasonic – detector distanta HC-SR04
* Modul senzor urmarire linie TCRT5000
* Placa de dezvoltare MEGA 2560 compatibil Arduino (CH340g)
* 2 x Placa de dezvoltare NANO V3 Atmega328p Arduino compatibil
* Senzor ultrasonic rezistent la apa
* Sursa alimentare 12V 3A
* 16 x Senzor Higrometru Capacitiv
* 3 x Modul Wireless transreciever NRF24L01 cu antena
* 4 x Modul Adaptor pentru nRF24L01
* Valva electromagnetica 12V, Apa/Aer, N/C, 1/2”
* 2 x Shield expansiune Arduino NANO
* Kit sasiu Smart Car 4WD
* Roata pivotanta robot
* Pompa Apa/Aer cu diafragma, 6-12V,R385
* Shield Modul L293D
* 2 x Servomotor SG90 360o continuu

**Authors:**

* Roșu Alin-Petru (6.1)
* Rășinar Ioan-Traian (6.1)
* Roșu Denisa-Rebeca (6.1)
* Volosciuc Cristian (7.2)