UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Faculty of Engineering, Built Environment and Information Technology

Fakulteit Ingenieurswese, Bou-omgewing en Inligtingtegnologie / Lefapha la Boetšenere, Tikologo ya Kago le Theknolotši ya Tshedimošo

Practical Guide

to an

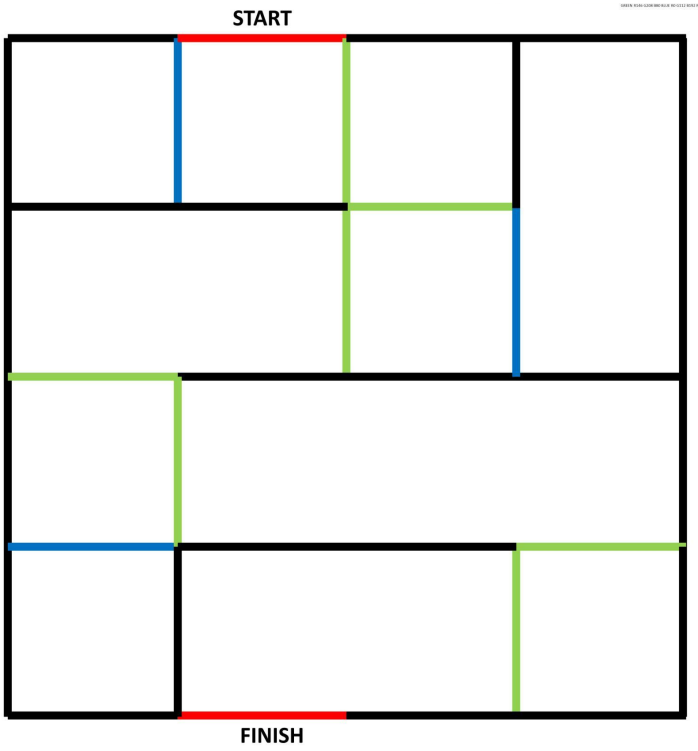
A-Maze-Eng MARV

ELO320/ERD320/EWE320

Version 2022.3 (Last update 11 August 2022)

Compiled by:

Dr W. Badenhorst| Prof. T. Stander | E.K. Stanley | B.D. Bauermeister | F. Kuschke | D. Strydom |M. Pretorius



INDEX

List of Abbreviations

EoC - End-of-Calibration
HMI - Human Machine Interface
LCD - Liquid Crystal Display
LED - Light Emitting Diode
MARV - Microcontroller-based Autonomous Robotic Vehicle
MDPS - Motor Driver and Power Supply
PEC - Physical and Environmental Considerations
SCS - System Communication Standard
SNC - State-and-Navigation Control
SS - Sensor Subsystem
SSD - Seven Segment Display

1. Introduction

The primary objective of the practical component of this course is for you to apply the Systems Engineering principles covered in the lectures by improving and converting your EMK310 MARV into an A-Maze-Eng MARV within a multi-disciplinary project team. You are required to design and construct a Microcontroller-based Autonomous Robotic Vehicle (MARV) capable of navigating through (not solving) a maze, such as the one on the cover page, using colour sensors, wheels attached to DC motors, a state machine and a predetermined set of navigation rules that will ensure your MARV does not get lost or stuck in the maze. Control of the MARV is achieved using microcontrollers of **any type** and programmed using **any language**.

The MARV will consist of three subsystems each targeted at a specific engineering discipline:

- a Sensor (SS) subsystem intended primarily for the electronic engineering student,
- a State-and-Navigation Control (SNC) subsystem intended primarily for the computer engineering student,
- a Motor Driver and Power Supply (MDPS) subsystem intended primarily for the electrical engineering student in the group.

Each of these subsystems must be able to communicate via serial communication with:

a) each other as an integrated system and
b) the HUB for individual assessment.

The HUB is a Python based emulation of the three subsystems which allows for individual, independent testing and evaluation of the subsystems. The latest version of the HUB software is available on ClickUP.

This document details the descriptions, architecture, design specifications and protocols required to develop the A-Maze-Eng MARV. These specifications must strictly be adhered to to ensure the correct operation of the MARV and compatibility with the HUB.

The functional and architectural block diagrams are given in Section 2 and Section 3 respectively which illustrate the functions and architecture of the system and subsystems and how they interact with one another. The software architecture is given in Section 4. The system specifications are given in Section 5 and provide the specifications that must be met by the system as a whole to ensure the desired operation. Section 6, Section 7, and Section 8 provide descriptions of each of the subsystems and the specifications each subsystem must adhere to. Section 9 gives a description of and the specifications for the Navigation Control. Section 10 briefly lays out the deliverables and evaluation criteria of the four practicals that will guide you toward delivering an A-Maze-Eng MARV. Appendix A contains the critically important system communication standard (SCS) that describes the communication protocol and the control byte structure that must be used when communicating via serial communication. Appendix B gives the physical and environmental considerations (PEC) of the system, including size considerations, units of measure and maze information. Appendix C lists important changes in this revision and Appendix D some potholes to be aware of during the implementation of the system.

2. Functional Block Diagram

The Functional Block Diagram, as seen in Figure 1, illustrates the required functionality for the complete MARV system, subdivided into the three main subsystems (i.e. Sensor (SS), State-and-Navigation Control (SNC) and Motor Driver and Power Supply (MDPS)). In the case of individual demonstrations or nonintegrated systems, the existing subsystem(s) will be connected to the HUB which will then emulate the missing subsystems as illustrated in Figure 1a. In the case of complete integrated systems, the HUB is excluded as illustrated in Figure 1b.

The **Sensor subsystem** (SS) is responsible for the calibration of all of the sensors in the sensor array (1.1), colour detection (1.2) and visualisation of the colour detected by each individual sensor in the array (1.3). Furthermore, the SS must calculate the incidence angle of the sensor array relative to lines (walls) of the maze when detecting a line (1.4) and detect the finish line or exit of the maze. The SS must be able to communicate and receive the prescribed information to and from the SNC/MDPS or HUB (1.5).

The **State-and-Navigation Control (SNC) subsystem** must communicate with the SS, MDPS and/or the HUB (2.5), and is responsible for enabling the system and initiating state transitions (2.1). The SNC must also be able to display critical system diagnostics obtained from the other subsystems (2.2) and facilitate the Human Machine Interface (HMI) through a touch sensor and a clap of the hands and snap of the fingers detection (2.3). The SNC must implement a navigation control algorithm (2.5) which enables navigation within the maze by communicating the necessary MARV movements to the MDPS subsystem where the motor control must be implemented (3.2).

The **Motor Driver and Power Supply (MDPS) subsystem** must ensure that the system can operate independently from wall socket power and supply the system with the prescribed voltage levels (3.1). The MDPS must also implement motor actuation and control to ensure the MARV rotates or moves in the direction and at the speed instructed by the SNC (3.2) which will require sensing the rotation, speed and distance (3.3) from the two motors used to drive and steer the MARV. The MARV rotation setpoints for the MDPS will be calculated in the SNC subsystem and communicated directly from the SNC or via the HUB (3.4). The MDPS must be able to communicate and receive the prescribed information to and from the SNC/SS or HUB.

All subsystem communication must comply with the protocol as stipulated in Appendix A.

As stated earlier, if all three degree programs are represented in a group, the recommended allocation of subsystems to disciplines is as follows:

1. *Electronic - SS subsystem*
2. *Computer - SNC subsystem*
3. *Electrical - MDPS subsystem*

However, the design task required for each of the three subsystems is suitable to be completed by a student from any discipline.

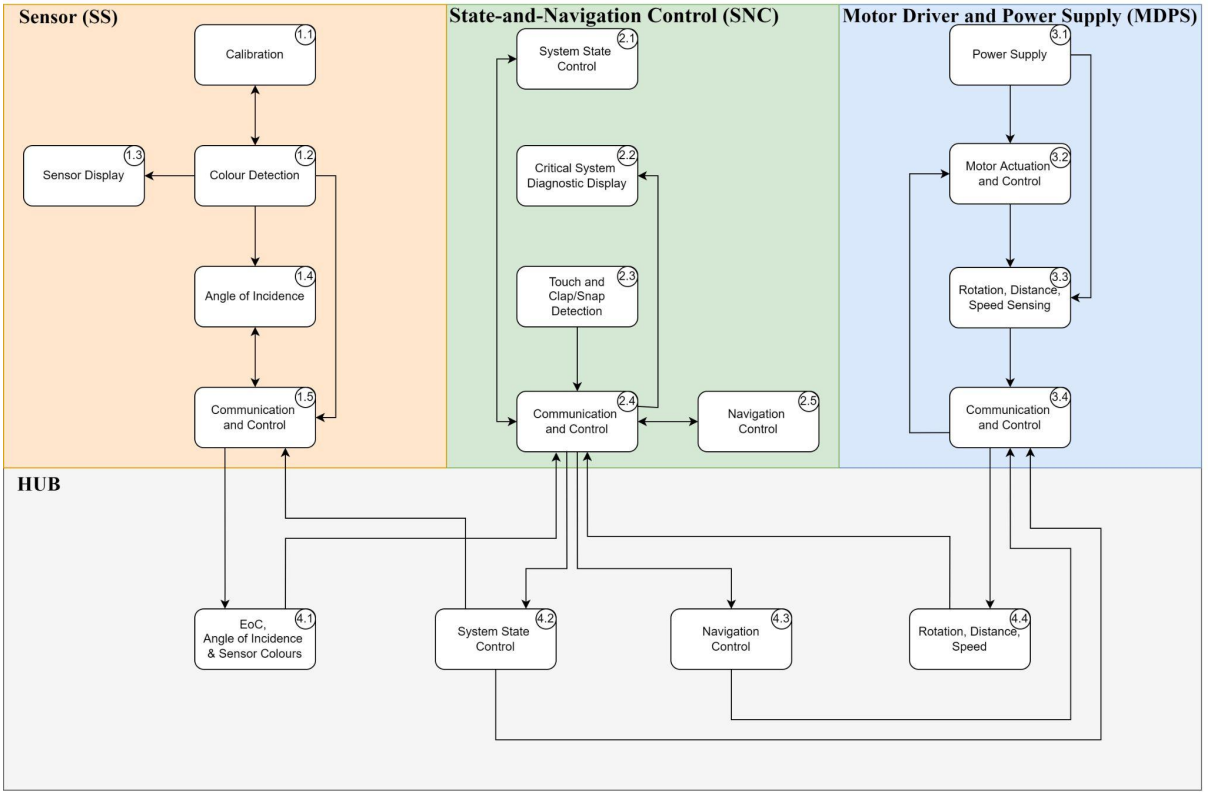


*Figure 1a - Prescribed system functional block diagram for **HUB integrated** system, which may use the HUB to emulate the two of the three subsystems and the environment.*

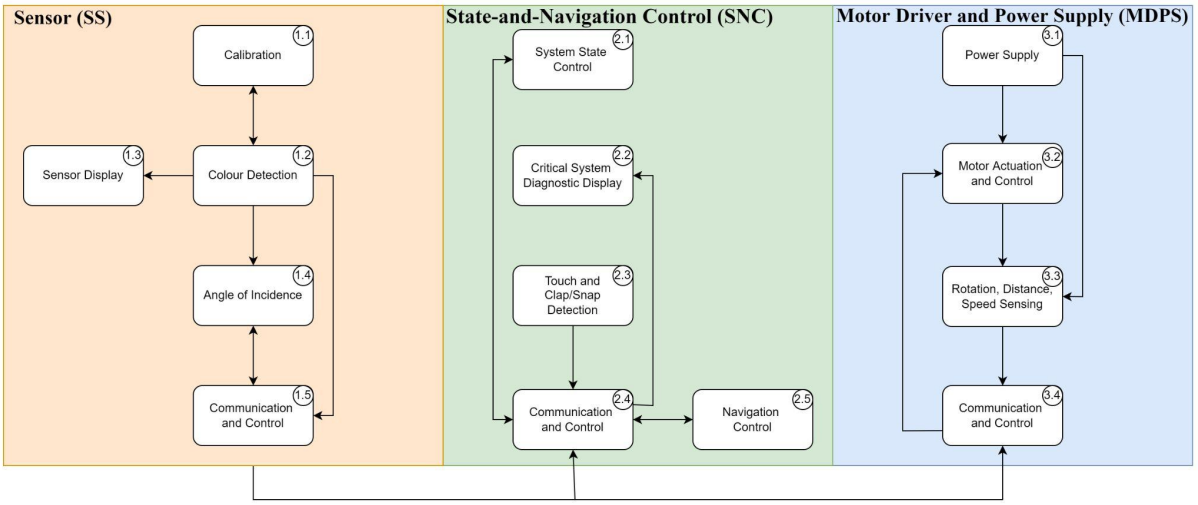


*Figure 1b - Prescribed system functional block diagram for **integrated** systems.*

3. Architectural Block Diagram

The Architectural Block Diagram shown in Figure 2 illustrates the required principal hardware components and subdivision of the entire system into the three main subsystems. Communication between the three subsystems (i.e. SS, SNC and MDPS) is done via serial UART. **Red** blocks represent off-the-shelf components, whereas **blue** blocks represent components that **must** be designed and implemented from first principles. First principles mean that you may **not** use an off-the-shelf module such as a microphone-filter-amplifier, or an H-bridge or an optical motor shaft encoder module. The detailed designs will form part of your final examination report. *Should students wish to implement any off-the-shelf components from first principles, they are allowed to do so.*

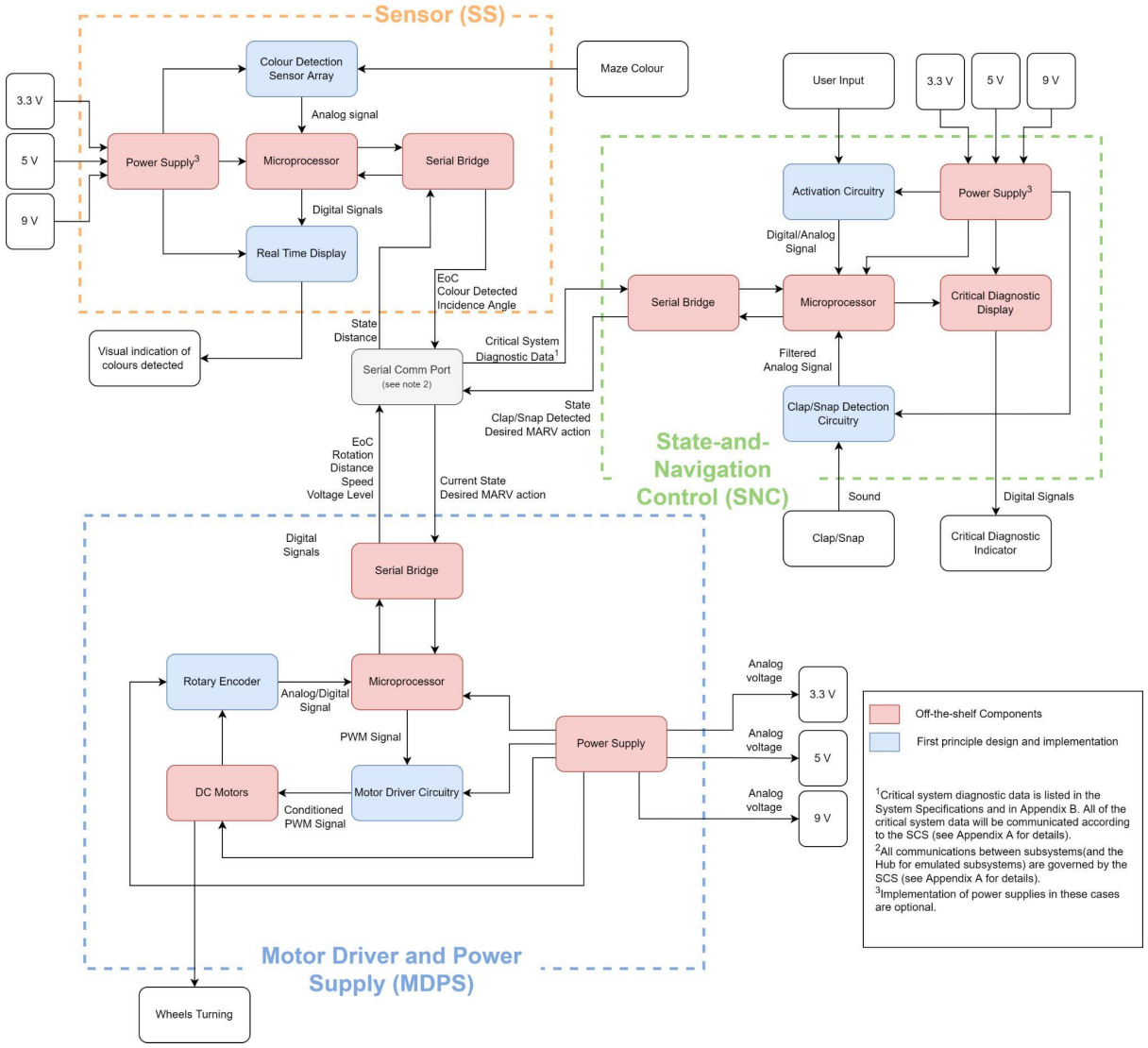


Figure 2 - Prescribed system architectural block diagram. Missing subsystems will be emulated by the HUB.

4. Software Architecture

The state diagram in Figure 3 describes the primary embedded control operating states of the system. Table 1 offers more detail about the expected outcomes for each of the control operating states of the system. A detailed version of the system state diagram, as well as the communication protocol, is provided in Appendix A.

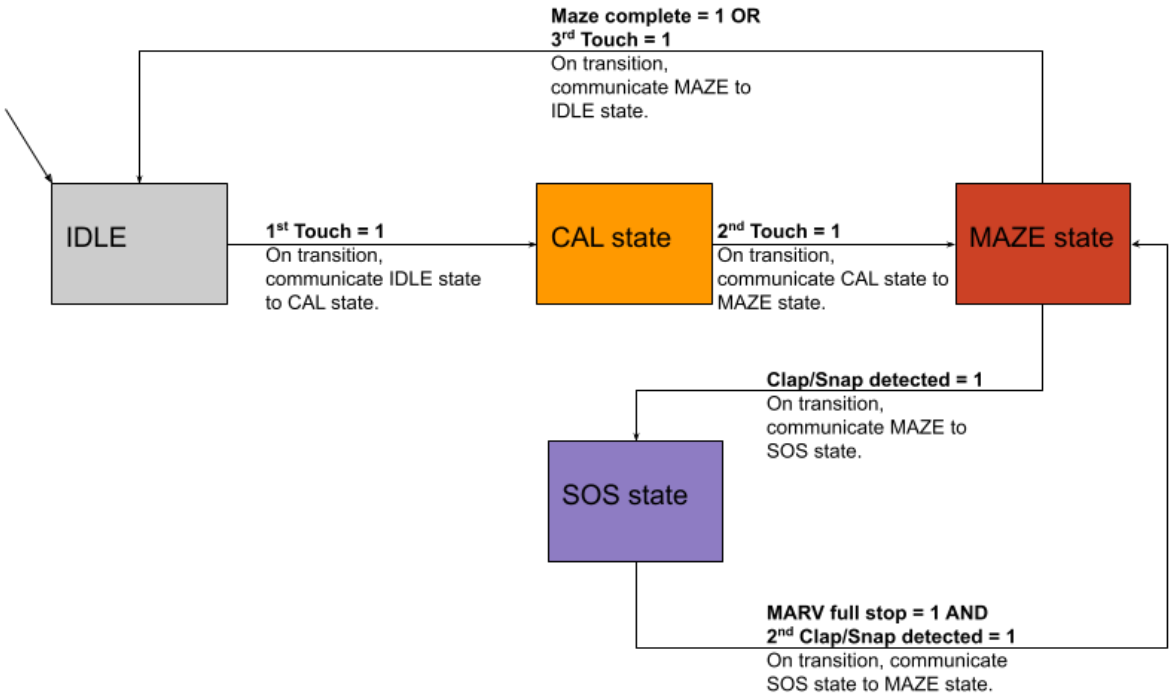


Figure 3 - Prescribed primary system operating states.

At power-up, the MARV has to enter the **IDLE state**. The system then waits in the IDLE state for user input which is the first touch activation meant to initiate the state transition from the IDLE state to the CAL state.

The colour sensor and motor sensor calibration is completed in the **CAL state**. After the calibration has completed, the SNC must display the available system diagnostics (as indicated in Figure 2 and listed in Table B.1 in Appendix B) while awaiting the second touch activation and state transition to the MAZE state.

In the **MAZE state** the subsystems are expected to facilitate the following system-critical functions to implement navigation.

1. The SS is responsible for detecting the colours (White, Red, Blue, Green, Black) and communicating the colour seen by each sensor to the SNC. Furthermore, it is responsible for calculating the incidence angle of the sensor array relative to a line when crossing it and communicating the angle to the SNC for navigation decision making.

2. The SNC is responsible for implementing the steering or navigation control algorithm using the information obtained from the SS. The SNC will calculate the required tangential wheel speeds and MARV actions (e.g. forward, backward, turn) to drive and navigate the MARV, and communicate this to the MDPS. Navigation control is detailed in Section 9. The SNC must also continuously display all critical system diagnostics and detect a clap of hands or snap of fingers in close proximity to enter or exit the SOS state.
3. The MDPS subsystem is responsible for implementing the motor control of the MARV to achieve the setpoints communicated by the SNC. The actual MARV rotation angle, speed and distance must be measured and communicated to the SNC for critical system diagnostic display. Distance refers to the distance the MARV travelled since the last stop or rotation. Wheel rotation sensing must be done with a rotary encoder using the wheel itself. Students will be provided with standardised 3D printed wheels and the stl-file to 3D print the wheels yourself, if needed be, that will fit on your EMK motor shafts. See Appendix B PEC for what the wheel looks like. Students may however use their own DC motor and wheel/encoder designs.

When a hand clap or finger snap is detected during the MAZE state, the system must transition into an emergency state referred to as the **SOS state**. While in the SOS state, the MARV is expected to come to a stop within a specified time and wait in the SOS state for a second clap or snap. The second clap/snap then triggers the transition back into the MAZE state. Upon completion of the maze, i.e. after having crossed the red line, the MARV must rotate 360° and then transition into the initial IDLE state.

Table 1: State outcomes and transition conditions

State ID	State Outcomes	Transition Conditions
IDLE	System awaits user input via touch activation through the touch-activated interface, which initiates a state transition to the CAL state. No output is expected on any of the other subsystems in this state.	The system must start in the IDLE state. The system must transition into the IDLE state from the MAZE state when the maze has been completed or if a touch activation is sensed the MAZE state.
CAL	The system performs calibration of the sensors in the SS and the MARV speed in the MDPS. After completion of the calibration sequence, the SNC subsystem is expected to display available system diagnostics (see Table B.1 in Appendix B for details on system diagnostics).	When the first touch activation is sensed in the IDLE state, the system must transition into the CAL state.
MAZE	The MARV must navigate through the maze (see Section 9). The SNC must display all critical system diagnostics (see Table B.1 in Appendix B for details on system diagnostics).	The system must transition from the CAL state into the MAZE state after the second touch activation OR from the SOS state after the second clap/snap detection.
SOS	The MARV is expected to come to a full stop within the specified time. The SNC must display all critical system diagnostics in this state. A second clap/snap detection must transition the system from the SOS state back into the MAZE state.	When a clap/snap is detected, the system must transition between the SOS and MAZE states.

5. System Specifications

1. The system architecture as depicted in Figure 2 must be implemented.
2. The system must implement the software state diagram as depicted in Figure 3 and Figure A.1 in Appendix A.
3. The system must adhere to the state outcomes and transitions as stipulated in Table 1 and illustrated in Figure 3.
4. The system must implement a communication and interface protocol between subsystems as stipulated in the SCS documentation (Appendix A).
5. The physical dimensions of the designed system must adhere to the specifications as stipulated in the PEC documentation (Appendix B).
6. The system needs to operate independently of wall socket power when navigating the maze, but may operate from a DC power supply when testing individual subsystems with the HUB.
7. The system must be able to operate continuously for at least 5 minutes in the MAZE state when using battery power.
8. The MARV must complete the practice maze (one on the front page) within 5 minutes, excluding time in the SOS state.
9. The system must adhere to the navigation specifications as described in Section 9b.

It is highly recommended to re-use components from previous courses, in particular EMK310. A limit of R 750 per team is set for the total additional cost that may be spent on the project.

6. State-and-Navigation Control (SNC)

a. Subsystem Description

The SNC subsystem is essentially the brain and ears of the MARV responsible for providing a human-machine-interface (HMI) and for controlling all state and navigation functions. HMI components required for the user to control the MARV and for the MARV to communicate with the user are:

- a touch-activated sensor to initiate the first state transition (i.e. from IDLE to CAL) and to transition from the MAZE to the IDLE state. This can be an app based activation, a capacitive touch sensor of some sort, but not a simple push button.
- a clap/snap sensor to transition between the MAZE and SOS states,
- a critical system diagnostic indicator or display.

The SNC is responsible for controlling and communicating all state transitions to the SS and MDPS or alternatively the HUB if the SS and/or MDPS is being emulated using the HUB. The SNC must receive and display all critical system diagnostic data from the SS and MDPS. Once in the MAZE state, the SNC is responsible for making the decisions to adhere to all the navigation specifications stated in Section 9b based on the SS data received and then communicating the desired motor action instructions to the MDPS.

It is compulsory that the subsystem's functionality and architecture are implemented as outlined in the Functional and Architectural block diagrams described in Figures 1 and 2 respectively.

b. Subsystem Specifications

The SNC must:

1. be able to function using one, or a combination of, the following input voltages, with a tolerated variance of 5%:
    a. 3.3 V
    b. 5 V
    c. 9 V
2. not draw more than 1 A from any of the three supply rails, resulting in a maximum allowable current draw from the power supply of 3 A.
3. adhere to the standard communications protocol as stipulated in the SCS.
4. manage and communicate all state transitions as indicated in Figure 3 and described in Table 1.
5. await end-of-calibration (EoC) signals from the SS and MDPS within the CAL state before displaying all the critical system data and allowing a 2nd touch of the touch sensor to transition the system from the CAL to the MAZE state.
6. make the decisions needed to meet the navigation specifications stated in Section 9b based on the SS data received and then communicate the desired motor rotation instructions to the MDPS.

7. detect a clap of hands and snap of fingers with a minimum volume of 70 dB.
8. receive and then display the following critical system diagnostic data using a web or smartphone application, an LCD screen or seven-segment displays (SSD), according to the display requirements as set out in the PEC documentation.
    1. Present system state: IDLE, CAL, MAZE or SOS.
    2. Last detected and recorded incidence angle in degrees.
    3. MARV distance covered in [mm] since the last stop or rotation.
    4. Tangential wheel speeds in [mm/s].
    5. Colour seen by each sensor in the sensor array.

    All information must be visible simultaneously.

7. Sensor Subsystem

a. Subsystem Description

The Sensor subsystem is essentially the eyes of the MARV with some intelligence built into it. The SS is responsible for detecting and distinguishing between the colours Red, Blue, Green, Black and White. The functionality and operation of the designed sensor circuit may be influenced by minor colour variations in the printed track colour, ambient lighting conditions or other unexpected external factors. Therefore, it is required that the output from the designed sensor circuit is subjected to a calibration state upon system startup (i.e. receiving a prompt from the SNC to enter the CAL state).

Upon receiving the command to enter the MAZE state, the responsibility of the SS is to provide the system with the necessary information to make decisions on how to navigate. This is achieved by communicating the colour seen by each sensor as well as the calculated incidence angle of the sensor array (i.e. MARV) to the SNC or HUB. To achieve this, the SS will require the distance travelled by the MARV since the last stop/rotation from the MDPS. For debugging and validation purposes, the SS is also required to physically visualise the colour that each sensor in the array is sensing in real time whilst the subsystem is operational (indicated in Figure 2).

It is compulsory for the subsystem's functionality and architecture to be implemented as outlined in the Functional and Architectural block diagrams described in Figures 1 and 2 respectively.

b. Subsystem Specifications

The SS must:

1. be able to function using one, or a combination of, the following input voltages, with a tolerable variation of of 5%:
    a. 3.3 V
    b. 5 V
    c. 9 V
2. not draw more than 1 A from any of the three supply rails, resulting in a maximum allowable current draw from the power supply of 3 A.
3. communicate with the SNC or HUB using the standard communication protocol as stipulated in the SCS.
4. have five (5) colour sensors in the sensor array. The sensors must be in a straight array, parallel to the wheel axis (refer to Figure B.1a in Appendix B).
5. be able to detect and classify the following colours as printed on the maze test block: White, Black, Red, Green, Blue. Refer to the PEC documentation and Figure B.2 for details and colour standards.
6. communicate the colour being detected by each sensor to the SNC or HUB when the calibration is completed.
7. visually depict the colour seen by each sensor in the array in real-time.

8. implement the states and transition between operating states as set out in the SCS.
9. calibrate within 60 seconds and send an end-of-calibration (EoC) signal.
10. use the two leftmost or two rightmost sensors to determine the angle of incidence (θ_i) within a $5°$ accuracy for $5° < \theta_i \leq 45°$ and communicate the result to the SNC or HUB adhering to the formatting requirements as stipulated in the PEC documentation.[1]
11. communicate the end-of-maze condition to the SNC or HUB when all the sensors detect the colour Red.
12. transmit no data to the SNC or HUB, while in IDLE or SOS state.

[1] If the incidence angle is > 45°, the MARV need not know what the angle is, just that it is > 45° and this the SNC can determine without a calculated angle.

8. Motor Driver and Power Supply Subsystem (MDPS)

a. Subsystem Description

The MDPS is the legs and the stomach of the system taking the MARV where the SNC tells it to go and providing the entire system with energy. The MDPS is responsible for supplying three prescribed, regulated voltage levels to the rest of the system and the monitoring of the unregulated battery level percentage.

In the MAZE state, the MDPS is responsible for controlling the MARV speed and direction as received from the SNC. It is also responsible for calculating the distance travelled since the last stop as well as the actual MARV rotation (turn) made in degrees. The MDPS must transmit the present speed, distance travelled since the last stop and degrees of the last rotation/turn to the SNC or HUB.

Upon receiving the SOS state transition notification from the SNC, the MDPS is responsible for stopping the MARV as specified below.

It is advised that optical isolation of the microcontroller is considered for a specific part in this subsystem. Furthermore, it is compulsory that the subsystem's functionality and architecture are implemented as outlined in the Functional and Architectural block diagrams described in Figures 1 and 2 respectively.

Power Supply Implementation:
The 3.3 V, 5 V and 9 V rails may be implemented with off-the-shelf buck converter modules or passive regulators.

b. Subsystem Specifications

The MDPS must:

1. operate independently of wall socket power when integrated with the other two subsystems.
2. output, externally, voltage levels of 3.3 V, 5 V and 9 V to within a 5 % accuracy under all load conditions, where the maximum external current drawn is 2 A for each rail while the motors are turning at nominal speed.
3. store enough energy to power the integrated system for at least 5 minutes in the MAZE state.
4. calibrate the tangential wheel speeds (v_{op}) in the CAL state within 60 seconds.
5. enable all necessary movements of the MARV to adhere to all the navigation specifications in Section 9b.
6. be able to propel the MARV at a speed v_{op} of at least[2] 10 mm/s.
7. ensure a speed to within 5% of the speed set by the SNC.
8. rotate around the midpoint between the two wheels of the MARV (see Figure B.1a).
9. allow rotation of the MARV of between 5° and 360° with an error of $< 5°$.

[2] The maximum speed allowed is 255 mm/s so as to transmit the measured speed in a single byte - refer to appendix A

10. determine the distance travelled since the last stop with an error ≤ 2.8 mm.
11. communicate with the SNC or HUB using the standard communication protocol as stipulated in the SCS.
12. communicate the following to the SNC or HUB, according to the display requirements in the PEC documentation:
    a. Tangential wheel speeds in [mm/s].
    b. Distance travelled since the last stop or rotation in [mm].
    c. Last rotation angle of the MARV in degrees.
13. come to a complete halt in less than 2 seconds upon entering the SOS state.
14. report no data to the SNC or HUB in the IDLE state.

9. Navigation Control (NAVCON)

a) Description

Refer to the cover page for an example of a maze. Black is a "wall", blue is a "closed door" (effectively equivalent to black), green is an "open door" and red is an "exit" or "entrance", though the MARV will start inside the "entrance". Why does the maze have blue lines if they are essentially treated as black? If we remove the blue lines, your MARV would not only have to navigate, but it would also need a maze solving algorithm. With the given navigation specifications, the blue lines prevent the MARV from getting lost in the maze and it tests if the sensors can detect blue. There are primarily four directions to consider when navigating.

- REVERSE: the direction that the MARV came from.
- FORWARD: the direction in which the MARV will be moving when encountering a line.
- RIGHT and LEFT rotation: As specified below, or incremental adjustments to make small course corrections if the MARV is not travelling perfectly parallel to a wall.

b) Specifications

The navigation control must be implemented as follows when in the MAZE state.

1. a) Not one of the wheels is allowed to ever cross a black or blue line and
   b) only two sensors are allowed to cross a black or blue line to determine the angle of incidence.
2. The MARV must be able to perform the following distinct turn functions:
   a. turn $90° \pm 5°$RIGHT
   b. turn $180° \pm 5°$ LEFT or RIGHT
   c. turn $360° \pm 5°$ LEFT or RIGHT
   d. an incremental rotation of $\leq 5°$ RIGHT AND LEFT (required for steering correction).
3. The MARV must stop *before* changing direction or turning (rotating).
4. The MARV may not traverse (cross) a green or red line with an incidence angle of $\theta_i > 5°$.
   If the MARV encounters a green line at an angle of $5° < \theta_i < 45°$, it must stop, reverse, apply steering correction, and re-attempt traversing the line.
5. If in going forward the MARV detects a green or red line at an incidence angle $\theta_i > 45°$, the MARV must reverse and apply steering correction through a single rotation of $\leq 5°$ **toward** the line. This process (forward, detect, reverse, rotate) must repeat until $\theta_i < 45°$ at which point specification 4 must be adhered to.
6. The following sequence[3] must be applied when encountering a black or a blue line at an incidence angle of $\theta_i < 45°$ after the MARV has been driving forward. The MARV must:
   a. Reverse,
   b. Apply a $90° \pm \theta_i$ RIGHT turn to move forward parallel to the black/blue line on its left,

[3] All mazes are set up such that following this sequence will navigate through the maze successfully.

c. Continue moving forward,
d. If, **after having turned right,** the MARV detects a green line OR no line, it must keep on driving forward,
e. If**, after having turned right,** the MARV, detects a blue OR a black line **BEFORE detecting a green line,** it must
    i. Reverse,
    ii. Apply a 180° turn,
    iii. Drive forward[4].

7. If in going forward the MARV detects a black or blue line at an incidence angle $\theta_i > 45°$ the MARV must reverse and apply steering correction through a single rotation of $\leq 5°$ **away** from the line. This process (forward, reverse, rotate) must repeat until the MARV steers clear of the wall (black) or closed door (blue).
8. After the MARV has crossed red (exit), the maze is solved and the MARV must rotate 360° and transition into the IDLE state.

[4] A RIGHT turn followed by a 180°turn effectively then constitutes a LEFT turn.

10. Practical Evaluation Schedule

The development of your A-Maze-Eng MARV will be evaluated in four stages or practical demonstrations spread through the semester with each counting the indicated % toward your semester mark. Deliverables and criteria provided are subject to change in which case sufficient notice will be provided. Refer to the course schedule for the practical dates.

Practical 1: Design and Simulation (5%)

Objective:

- Individual evaluation of the design and simulation (where appropriate) of the blue architectural blocks and relevant software flow diagrams within each of the three subsystems as shown in Fig. 2, i.e. the system components as per the "*Hierarchy of Complex System*" in your textbook (study unit 1.2).
- You must present all your designs and calculations, your simulation or software test bench results as well as the comparison of the simulation / test bench output to your theoretical design outcomes in a **hardcopy** lab book. Herewith a quick video example of what we would typically like to see in your lab book.

Modus operandi:

- There will be two evaluation teams of three ALs/demis each evaluating two groups at a time.
- All three subsystems/students will be evaluated at the same time by one of the ALs/demis in the evaluation team.
- The duration of the evaluation will be only 5-6 minutes, therefore you must have your lab book with the designs and simulation results ready and the simulations up and running.

Deliverables:

The following will be evaluated for each subsystem with marks indicated in [].

MDPS [8]

- Driver and H-bridge design [1] and simulation [1].
- Rotary encoder design [1] and simulation [1].
- Speed and distance calculation algorithm [1] and test bench [1].
- Lab book [2].

SENSOR [8]

- Sensor array design [1] and simulation [1].
- Display design [1] and simulation [1].
- Angle of incidence calculation algorithm [1] and test bench [1].
- Lab book [2].

SNC [8]

- NAVCON design [1] and test bench [1].
- Clap/Snap sensor design [1] and simulation [1].
- Activation circuit design [1] and simulation [1].
- Lab book [2].

Evaluation criteria:

The design and simulation / test bench result of each system component as well as any flow diagrams will be evaluated separately using a three-point scale:

1 = Good to excellent, more than 75% done.

0.5 = Acceptable/sufficient with some shortcomings, heading in the right direction.

0 = Poor design/simulation/testing/logic or less than 25% done .

The lab book, counting two marks, will be evaluated on how well or with what ease another student can use the lab book to follow, understand and reproduce what has been designed and simulated.

Note: A circuit diagram in itself is not a design but primarily a depiction of the result of the design process. Your design must show how you got the resistor values in your circuit diagram, motivate your chosen topologies and parts. Why is that resistor 10 kΩ? How did you decide on that particular transistor for your H-bridge? Why did you implement an array, rather than separate distinct variables, in your code?

Practical 2: Developmental tests (5%)

Deliverables:

- Students must present the latest simulations of their system components and indicate how the design has been improved since Practical 1 - everything must be in your lab book.
- A demonstration of the system components prototypes. These may still be on a breadboard.
    - Students must design and demonstrate their own developmental tests (study units 1.3 and 3.1) for their system components to show the evaluators how well the prototypes compare with the simulations.
- All subsystems must demonstrate successful communication with the HUB as per the SCS in Appendix A. The following calibration QTPs will be evaluated using the HUB for each of the three subsystems.
    - SNC - QTP2
    - SS - QTP2
    - MDPS - QTP3

  For this demonstration you may hardcode valid data packets <dat1,dat0,dec> where hardware has not yet been implemented.

Evaluation criteria:

- The same scale used in Practical 1 will apply.
- For each system component:
    - Is this a valid/good/relevant developmental test?
    - Is the test design acceptable?
    - What is the quality of the test outcomes and how well does it compare to the simulation outcomes?
- Can the subsystem successfully transmit to and receive data from the HUB as per the SCS?

Practical 3: Individual QTP demonstration with the HUB (20%)

As the weight of this practical indicates, this is the most critical of all the practical demonstrations and, as per the the subminimum requirements in your study guide, requires a mark of at least 50% for advancement to Practical 4 and admission to the examination.

Deliverables:

- Students must demonstrate at least two of their subsystem's QTPs (excluding QTP1 for all subsystems) with the HUB.
- Failure to communicate or demonstrate with the HUB will result in an immediate zero mark.
- This demonstration has a subminimum for exam entry associated with it and therefore students will have the opportunity to re-demo should they fail the practical. The re-demo mark will be capped at 50%.

Evaluation Criteria:

- Detailed evaluation sheets will be made available on ClickUP, but below are a few examples of the format. CnD = “Could not Demonstrate”

Group	Student No:	SNC / NAVCON [26]
	Name:	

	NAVCON QTP 1: Traverse Green/Red at $\theta_i < 5°$ [6]	Yes	Partly	No	CnD	N/A	Marks
N1.1	The MARV must stop before changing direction or rotating.						2
N1.2	The MARV must cross RED or GREEN at $\theta_i \leq 5°$ when the line is encountered at $5° < \theta_i < 45°$.						2
N1.3	The MARV must cross RED or GREEN without rotating when the line is encountered at $\theta_i \leq 5°$.						2

	SNC QTP 2: Touch sensor state transition and critical diagnostic display [7]	Yes	Partly	No	CnD	N/A	Marks
S2.1	The SNC must display only IDLE at startup						1
S2.2	The SNC correctly transitions to and indicates the CAL state after the 1st touch sensor activation.						1
	The display correctly indicates the CAL state and correctly displays						

	MDPS QTP 3: Calibration and Communication [6]	Yes	Partly	No	CnD	N/A	Marks
3.1	The MDPS must not transmit in IDLE state.						1
3.2	The MDPS must calibrate the wheel speeds to the speeds entered into the Hub GUI within 5% in both test runs.						2
3.3	The MDPS must function and communicate as stipulated in the SCS state diagram.						1
3.4	It must be proven that the second test run executed at a lower speed.						2

	SS QTP 4: Incidence Angle [6]	Yes	Partly	No	CnD	N/A	Marks
4.1	The SS measures and transmits an incidence angle within a 5° accuracy for $5° < \theta_i < 45°$ on the **RIGHT.**						3
4.2	The SS measures and transmits an incidence angle within a 5° accuracy for $5° < \theta_i < 45°$ on the **LEFT.**						3

Practical 4: Integrated system demonstration (15%)

Deliverables:

- Fully integrated systems capable of navigating through a maze.
- In cases where students did not pass practical 3 and are excluded from Prac 4 resulting in incomplete groups, students will, where possible, be regrouped to form complete systems. Where not possible and the group only has two members (2 subsystems), both subsystems will again be evaluated individually using the HUB and then the group must, at bare minimum, demonstrate the following to prove that the two subsystems can communicate with each other as per the SCS.
    - Put both subsystems in MAZE mode using the SCS.
    - Physically connect your two subsystems together so that it can communicate via the

SCS.

- If you have the SS and SNC in your group:
  - Via a terminal (putty, termite etc) send the last MDPS packet (c224) to the SS.
  - The SS must then transmit its two packets as per the SCS state diagram.
  - The SNC must then follow by transmitting its three packets as per the SCS state diagram.
- If you have the SNC and MDPS:
  - Via a terminal, send the two SS packets (c231 and c2232) to the SNC.
  - The SNC must then transmit its three packets as per the SCS state diagram.
  - The MDPS must then follow by transmitting its three packets as per the SCS state diagram.
- If you have the MDPS and SS:
  - Via a terminal, send the last SNC packet (c213) to the MDPS.
  - The MDPS must then transmit its four packets as per the SCS state diagram.
  - The SS must then follow by transmitting its two packets as per the SCS state diagram.
- All the transmissions must be visible in the format of control – dat1 – dat0 – dec on the terminal.

Evaluation criteria

- Integrated systems will be evaluated based on their performance in navigating the maze. The further the Marv progresses through the maze before making an error, the more marks you will score.
- Incomplete systems will be evaluated as described above.

Appendix A

System Communication Standard (SCS)

The subsystems communicate with each other (and the HUB when a subsystem is missing or inoperational thus being emulated by the HUB) using UART by sending data packets. Each packet consists of 1 control byte and 3 data bytes that when combined, forms a single 32 bit DWORD or 4 byte data PACKET = <SYS:SUB:IST:DAT1:DAT0:DEC> with the structure as shown below.

The serial communication has the following setup:

- **Baud Rate** = 19200.
- **Asynchronous** transmission/reception
- **8-bit** transmission/reception mode

Data Packet: Packet contains 1 control byte and 3 data bytes. (R/W = Readable/Writable)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SYS<1:0>		SUB<1:0>		IST<3:0>			
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DAT1<7:0>							
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DAT0<7:0>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DEC<7:0>							
bit 7							bit 0

bit 31-30	**SYS:** System state 11 = 3 = SOS_STATE; System is in emergency mode. 10 = 2 = MAZE_STATE; System is in MAZE mode. 01 = 1 = CAL_STATE; System is in calibration/monitor mode. 00 = 0 = IDLE_STATE; System is in idle mode.
bit 29-28	**SUB:** Subsystem number sending the packet 11 = 3 = Sensor subsystem. 10 = 2 = MDPS subsystem. 01 = 1 = SNC subsystem. 00 = 0 = HUB (reserved).
bit 27-24	**IST:** Internal state of the subsystem sending the packet 1111 = Substate 15. … 0011 = Substate 3. 0010 = Substate 2. 0001 = Substate 1. 0000 = Substate 0.
bit 23-16	**DAT1:** Upper date byte of the transmitted data.
bit 15-8	**DAT0:** Lower data byte of the transmitted data.
bit 7-0	**DEC:** Decimal or general purpose data byte of the transmitted data.

The system and subsystems start operation in IDLE mode and then sequentially move from one state to the next as shown in the state flow diagram in Figure A.1 on p25. If the HUB is used, it will transmit an all-zero control byte at which point the SNC starts detection of the touch-activated sensor. The rules of the SCS are as follows.

1. All three subsystems must be connected on the same transmit and receive channel to keep track of the system and internal subsystem states and react accordingly. The control byte is used to determine the state of the system (SYS) and the internal state of a subsystem (IST).
2. Only one subsystem is allowed to transmit at a time while the other two receive and interpret what has been sent.
3. When and what a subsystem is allowed to transmit is dictated by the control byte received as depicted in the state flow diagram in Figure A.1.
4. Each subsystem will send the control byte to indicate the transition/completion of its function along with the necessary 3 data bytes. The HUB will only send serial packets for the subsystems it is emulating.
5. Only once a subsystem has finished sending its data packet, is the system and subsystems allowed to transition to the next state.

As an example, let us go to Figure A.1 and assume the system is in the MAZE state and that the SNC has just indicated to the system that it is now in NAVCON internal state by transmitting the control byte <SYS=2/SUB=1/IST=3> = <10010011> along with a required MARV speed of 0. The control byte indicates to the SS that it is not allowed to transmit now and to the MDPS that it may now transmit

the battery voltage level (COM_OUT = LEVEL). (*As from 2022 this component has been removed from the MDPS subsystem, but it is kept in the state diagram otherwise the HUB will require significant revision. The MDPS must therefore simply send all data bytes <DAT1:DAT0:DEC> as zeros as indicated in the SCS.*) This does not mean however that the MDPS must immediately transmit the voltage level (zeros). Since the SNC's transmission showed the MARV's speed must be set to zero, i.e. stop the MARV, the MDPS could immediately upon receipt of the SNC's transmission, adjust the PWM to the two motors so as to stop the motors and *then* move to IST_STATE = LEVEL to transmit the rotation angel followed by the IST_STATE = ROTATION. But before the MDPS moves to IST_STATE = SPEED to transmit the speed of the MARV, it should wait until the sensors confirm that the MARV has come to a standstill and only then does the MDPS need to go into IST_STATE = SPEED and transmit the speed of the MARV. All this time the SS simply receives and monitors the communication until the MDPS finally sends the control byte <SYS=2/SUB=2/IST=4> = <10101000> indicating to the SS that it can now either go into the End-Of-Maze or COLOURS internal state. The SS must however continue to display the five sensor colours on its onboard display.

Following Figure A.1 is the control command library table containing all the necessary details regarding each subsystem's internal state within each of the four primary system states. It details the control byte command instructions, the structure and content of the DAT1, DAT0 and DEC bytes as well as descriptions of each internal state.

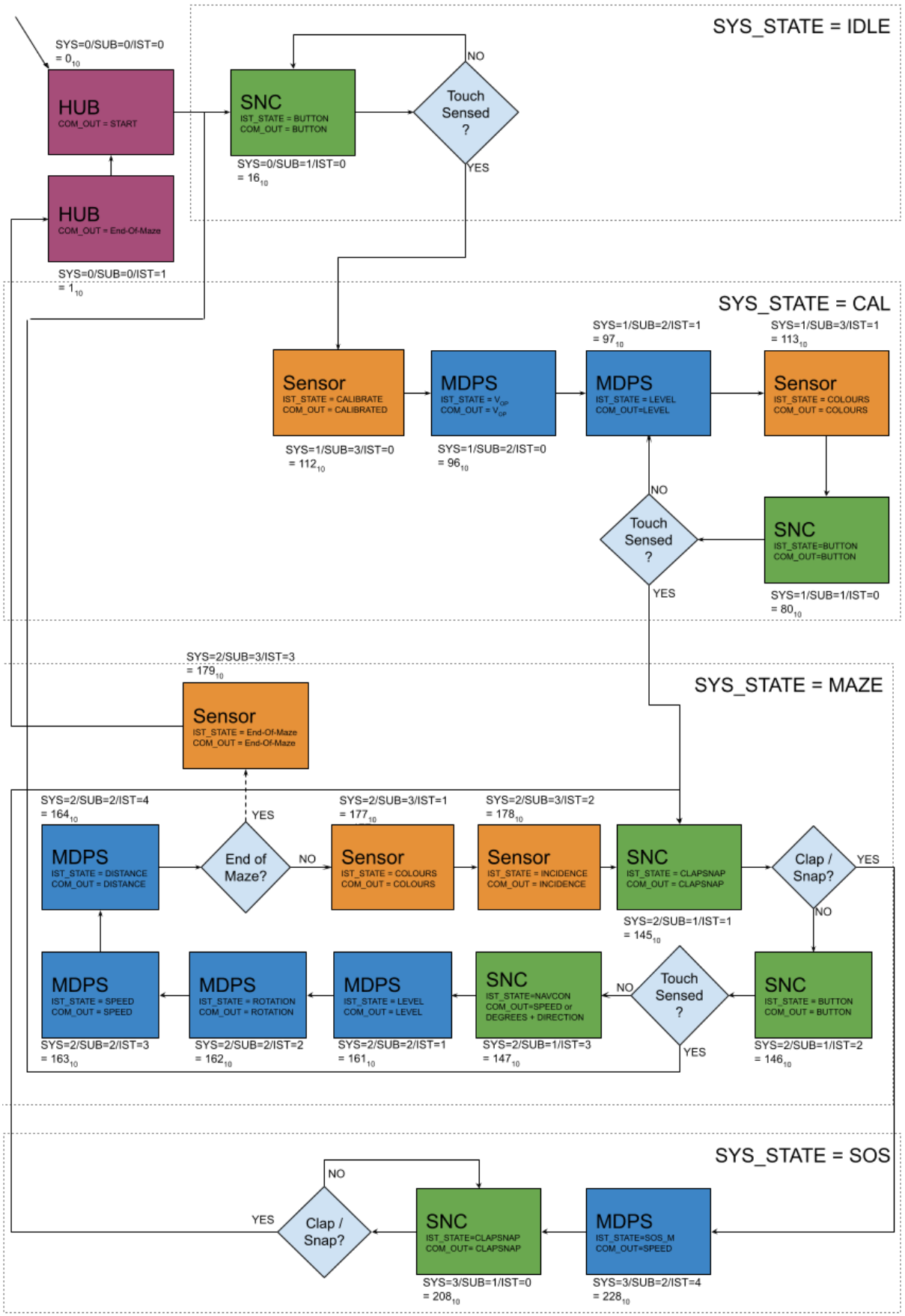


Figure A.1 - MARV SCS State Diagram.

SCS: Control Command Library (x = unknown/don't care).

	Control Byte <31:24>					
Subsyst.	**SYS <1:0>**	**SUB <1:0>**	**IST <3:0>**	**Control Command Description**	**Control Action**	**Additional Notes**
HUB	0	0	0	*HUB.* Initiate system (start) command.	System and Subsystems move to IDLE state.	SNC starts touch detection in IDLE state.
	0	0	1	*HUB.* Terminate system (stop) command.	System and Subsystems move to End of maze state.	
	0	0	x	*HUB.* Reserved/ Not in use.	None.	
Subsyst.	**SYS <1:0>**	**SUB <1:0>**	**IST <3:0>**	**Control Command Description**	**Control Action**	**Additional Notes**
SNC	0	1	0	*SNC.* Touch Detected (IDLE).	DAT1 = 1 if touch detected, else DAT1 = 0. DAT1 = 1: System and Subsystems move to CAL state. DAT1 = 0: Remain IDLE	
	0	1	1	*SNC.* Reserved/ Not in use.	None.	
	0	1	2	*SNC.* Reserved/ Not in use.	None.	
	1	1	0	*SNC.* Touch Detected (CAL).	DAT1 = 1 if touch detected, else DAT1 = 0. DAT1 = 1: System and Subsystems move to MAZE state. DAT1 = 0: Remain in CAL	DAT1 contains whether a touch was detected. DAT1 = 1: SNC moves to Clap/Snap detection in MAZE state.
	1	1	1	*SNC.* Reserved/ Not in use.	None.	
	1	1	2	*SNC.* Reserved/ Not in use.	None.	
	2	1	0	*SNC.* Reserved/ Not in use.	None.	

	2	1	1	*SNC.* Clap/Snap Detection (MAZE).	DAT1 = 1 if Clap/Snap was detected, else DAT1 = 0. DAT1 = 1: System moves to SOS state DAT1 = 0: SNC moves to Touch Detection	DAT1 contains whether a clap or snap was detected.
	2	1	2	*SNC.* Touch Detection (MAZE).	DAT1 = 1 if touch detected, else DAT1 = 0. DAT1 = 1: System and Subsystems move to IDLE state. DAT1 = 0: SNC moves to navigation control state (NAVCON)	DAT1 contains whether a touch was detected. DAT1 = 1: SNC moves to Touch Detection in IDLE state *after clearing DAT1*.

	2	1	3	*SNC.* Navigation Control (NAVCON).	DEC = 0 or 1: DATA bytes contain tangential wheel speeds.
					0 = forward
					1 = backward
					
					DEC = 2 or 3: DATA bytes contain required angle of rotation of the MARV
					2 = left (CCW, positive).
					3 = right (CW, negative).
						if DEC = 0 or 1:
						DAT1 = right wheel speed in mm/s,
						DAT0 = left wheel speed in mm/s
						
						if DEC = 2 or 3:
						DATA = <DAT1:DAT0> contains the angle of rotation in degrees between 5° and 360°.

	3	1	0	*SNC.* Clap/Snap Detection (SOS).	DAT1 = 1 if Clap/Snap, else DAT1 = 0. DAT1 = 1: System and Subsystems move to MAZE state.	DAT1 = 1: Clap/Snap was detected. SNC moves to MAZE Clap/Snap detection *after clearing DAT1*. DAT1 = 0: Wait/loop
Subsyst.	**SYS <1:0>**	**SUB <1:0>**	**IST <3:0>**	**Control Command Description**	**Control Action**	**Additional Notes**
MDPS	0	2	0	*MDPS.* Reserved/ Not in use.		
	0	2	1	*MDPS.* Reserved/ Not in use.		
	0	2	2	*MDPS.* Reserved/ Not in use.		
	0	2	3	*MDPS.* Reserved/Not in use.		

	1	2	0	*MDPS.* v_{op}. (CAL) v_{op} is the default forward tangential wheel speed.	DATA bytes contain measured tangential speeds of the wheels. Control packet sent a*fter* calibration is complete.	DAT1 = right wheel v_R in mm/s, DAT0 = left wheel v_L in mm/s.
	1	2	1	*MDPS.* Battery Voltage Level (CAL).	DAT1 = 0 DAT0 = 0 DEC = 0	Battery voltage level sensing removed in 2022. Send 0's
	1	2	2	*MDPS.* Reserved/ Not in use.		
	1	2	3	*MDPS.* Reserved/ Not in use.		
	2	2	0	*MDPS.* Reserved/ Not in use.		
	2	2	1	*MDPS.* Battery Level	DAT1 = 0 DAT0 = 0 DEC = 0	Battery voltage level sensing removed in 2022. Send 0's
	2	2	2	*MDPS.* MARV measured ROTATION	DATA = <DAT1:DAT0> contains the last measured executed rotation angle. DEC = 2: Rotation was to the left (CCW, positive). DEC = 3: Rotation was to the right (CW, negative).	The executed rotation angle refers to the actual measured angle the MARV rotated after having received a rotation instruction from the SNC. SNC displays the angle on indicators.
	2	2	3	*MDPS.* MARV Measured SPEED	DATA bytes contain measured tangential speeds of the wheels.	DAT1 = right wheel speed in mm/s, DAT0 = left wheel speed in mm/s. SNC displays the speeds on indicators.
	2	2	4	*MDPS.* MARV Measured DISTANCE moved since last stop/rotation.	DATA = <DAT1:DAT0> contains measured distance in mm.	It is the distance the MARV travelled in a straight line to the nearest mm.
	3	2	4	*MDPS.* Clap/Snap response.	Motor reduces speed to zero.	After stopping, DAT1 = 0 right wheel speed DAT0 = 0 left wheel speed.

Subsyst.	SYS <1:0>	SUB <1:0>	IST <3:0>	Control Command Description	Control Action	Additional Notes
Sensor	0	3	0	*Sens.* Reserved/ Not in use.		
	0	3	1	*Sens.* Reserved/ Not in use.		
	1	3	0	*Sens.* End of Calibration	MDPS moves to speed calibration	
	1	3	1	*Sens.* COLOURS (CAL)	DATA = <DAT1:DAT0> contains colour sensed by each sensor as a 3 bit code: W = White = 000 R = Red = 001 G = Green = 010 B = Blue = 011 K = Black = 100	DATA<2:0> = sensor 5 DATA<5:3> = sensor 4 DATA<8:6> = sensor 3 DATA<11:9> = sensor 2 DATA<14:12>=sensor 1
	2	3	0	*Sens.* Reserved/ Not in use.		
	2	3	1	*Sens.* COLOURS (MAZE)	DATA = <DAT1:DAT0> contains colour sensed by each sensor as a 3 bit code: W = White = 000 R = Red = 001 G = Green = 010 B = Blue = 011 K = Black = 100	DATA<2:0> = sensor5 DATA<5:3> = sensor 4 DATA<8:6> = sensor 3 DATA<11:9> = sensor 2 DATA<14:12>=sensor 1
	2	3	2	*Sens.* INCIDENCE	DAT1 contains the last measured angle of incidence.	SNC displays the last measured angle of incidence in degrees on indicators.
	2	3	3	*Sens.* End-Of-Maze (EOM)	End of maze detected.	SNC/HUB transmits EOM control byte. All subsystems move to IDLE state.

Appendix B

MARV Physical and Environmental Considerations (PEC)

Physical dimensions of the MARV

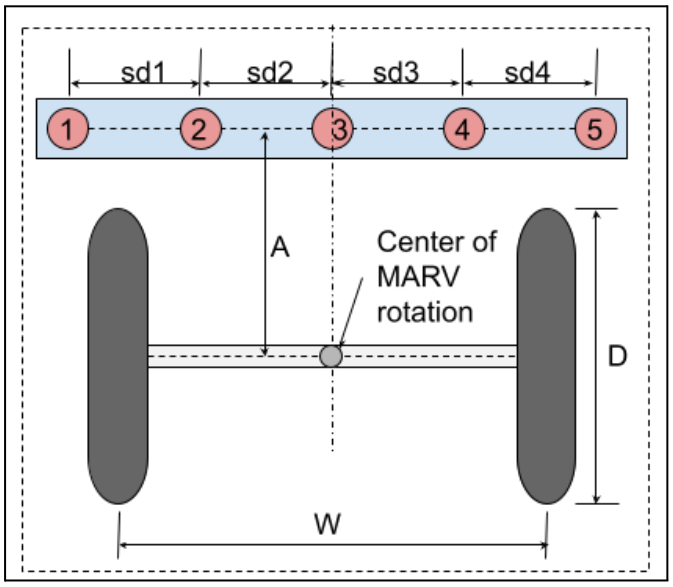
Each of the 16 blocks in the maze is 21 cm x 21 cm excluding the lines that are 1 cm thick. The width and length of the MARV therefore needs to be designed for it to be able to move and navigate within these dimensions. The HUB will require the following dimensions of your design as illustrated in Figure B.1a below.

From the SS:

- Distance between each of the sensors in the sensor array (sd1, sd2, sd3, sd4). *These distances need NOT be equal*, however, the middle sensor (no. 3) *must* be positioned on the centre line of the MARV to serve as a point of reference for the HUB.
- Distance between the wheel axis and the sensor array axis: axial distance (A).

From the MDPS:

- Distance between the wheels or the wheel base width (W)
- Outer wheel diameter (D)



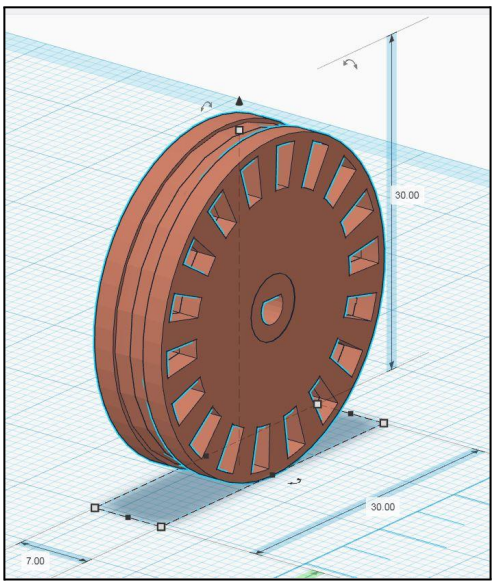


Figure B.1a) Top view of MARV dimensions required by the HUB indicating inter-sensor distances sd1 to sd4, inter axial distance A, wheel diameter D and wheel base width W.

b) Wheel design to fit the EMK motor shaft with 18 encoder slots and a groove for rubber bands. Outer wheel diameter = 30 mm, width = 7 mm.

Note: Figure B.1a is just for illustrative purposes and your relative dimensions and distances will of course be different.

Critical System Diagnostics

The information to be displayed by the SNC for each of the critical system diagnostic values is summarised in Table B.1.

Table B.1 - Critical system diagnostic data display summary.

Critical System Value	**Display Information**
Sensor Colours	Simultaneously display the colour detected (seen) by each of the five sensors. The system must distinguish between, and display notifications for, a minimum of five discrete colours using at least the following abbreviations: 1. W - White 2. R - Red 3. G - Green 4. B - Blue 5. K - Black *Should the student wish to implement additional colour indicators, e.g. colour indicators on an app or the colours written in text, they are allowed to do so.*
Present State	Display the four system states as: 1. IDLE 2. CAL 3. MAZE 4. SOS
MARV rotation angle	Display the MARV's last measured executed rotation angle in degrees rounded to the nearest degree.
MARV speed	Display the MARV's individual wheel tangential velocity in units of [mm/s] rounded to the nearest integer.
MARV distance	Display the distance the MARV travelled since the last stop or rotation in [mm] rounded to the nearest integer.
Incidence angle	Display the last recorded angle of incidence in degrees rounded to the nearest degree.

Maze test block

Each group will receive a test block as shown in Figure B.2 printed on a glossy, banner PVC. The block can be used to test all the MARV specifications. Please note that this block is on loan only and must be returned at the end of the semester. The mazes are printed on the same PVC material. You can download a pdf version of the test block, but remember that your sensors will act very differently on paper than on PVC. Final demonstrations will be conducted on the complete mazes printed on PVC. The test maze will be made available during certain of the scheduled practical sessions.

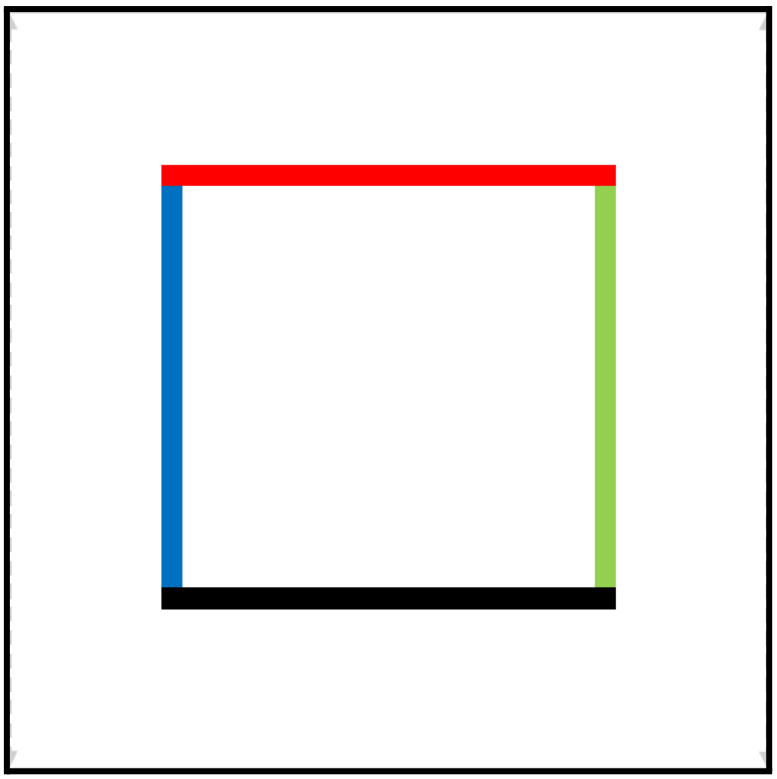


Figure B.2 The maze test block (scale 1:4)

Appendix C

Important Revision Changes

2022v2

- Removal of the required MDPS voltage sensing circuit and display in the functional block diagram (Fig 1a & b) and the architectural block diagram (Fig 2). The voltage level is however kept in the SCS state diagram to prevent significant revision of the HUB.

2022v3

- Revised Section 10, Practical 1 description.

Appendix D

Potholes

1. Serial comms with an Arduino

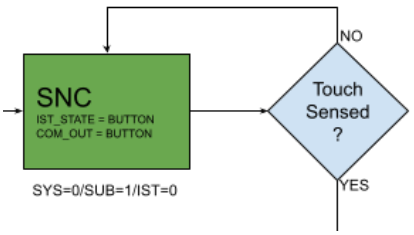
An Arduino does the following when sending a byte using the **serial.print(value)** function.

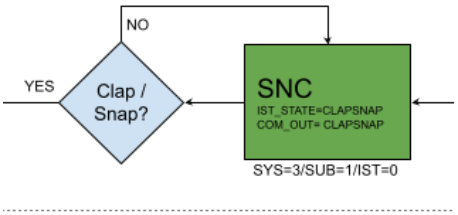
- It assumes 'value' is in string format.
- It then takes each character in 'value' and converts it into an 8 bit ASCII value.

For e.g. you want to send decimal 30 which is 0x1E via **serial.print(1E)** or **serial.print(30,HEX).**
It then converts the 1E (seeing it as a string) into a '1' in ASCII 8 bit value and an 'E' in ASCII 8 bit value and then transmits it as two separate bytes. So for every 1 byte you think you are transmitting, you are actually transmitting 2 bytes. The same principle applies using **serial.read()**, but just in reverse.

2. Spamming the HUB

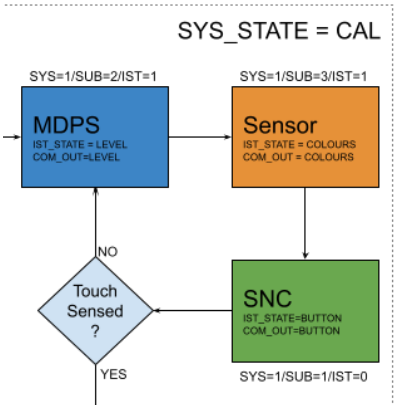
Do not "spam" the HUB when in a loop such as in the state diagram sections shown below.





For example, you will send c010-0-0-0 to indicate a touch has not been sensed in CAL state. There is then no need to continuously resend c010-0-0-0 every few microseconds or milli-seconds while waiting for a touch to be made. You could for example only resend every second or even only send c010-1-0-0 once a touch has been sent. You must however send at least one c010-0-0-0 otherwise the HUB will think your SNC is dead and not transmitting.

However, in the case below, each subsystem must send its required 4-byte packet every time the preceding subsystem in the state flow has sent its 4-byte packet.



3. Do not send a STOP instruction to the system after a ROTATE instruction.

The HUB will give an ERROR message if you do. Remember that the MDPS will by default be standing still after a rotation and hence no need to tell it to stop if it is already standing still.

4. **When the MDPS should reset DISTANCE to zero**

   The definition given for the MARV distance is “the distance the MARV travelled since the last stop or rotation”.

   - First, the MARV must stop before rotating or changing direction as stated in the specifications.
   - When the MDPS receives a STOP instruction from the SNC (c213-0-0-0), the MDPS will transmit LEVEL and ROTATION
   - While the MDPS is transmitting LEVEL and ROTATION, the MARV might already be coming to a standstill and only once it has stopped, should the MDPS transmit SPEED = 0.
   - The DISTANCE to be transmitted is the distance travelled since the ***last*** stop, i.e. not the *current* stop. In other words, you must first transmit DISTANCE (which will be the distance between the *last* and *current* stops) and only *then* reset DISTANCE to zero to start counting again once the MARV starts moving forward or backwards.

   The above manner is the way in which the emulated MDPS operates.

5. **Spacing of the 5 sensors**

   Using the same sensor array used in EMK310 or EBB320, i.e. small inter-sensor distances for following a line, will not work for this system. You must design the inter-sensor distances based on the requirements and specifications.