SDU

Assignment 1

Bee-Bot

by: Team (2/Thursday: 08-12) 7
Michalina Janik & Michał Krukowski,
mikru21@student.sdu.dk,
mijan21@student.sdu.dk

Hardware:

1. 2 x DC motors, L298 dual H-bridge IC

To control two DC motor, we are using L298 dual H-bridge IC built in the Arduino Motor Shield Rev3 which has the features of L293D but also can measure the current flow.

Used Pins and its functions:

Function	Direction	Speed	Brake
Channel A	12	6*	9
Channel B	13	11	8

We have switched the pin 3 to pin 6, due to the fact that we need pin 3 to the Interrupted function.

2. Multiplexer CD4051BE

Multiplexer is used to spare number of used hardware pins for several channels that can be addressed only by four of them (A, B, C, COM OUT/IN).

To Channels IN/OUT we have connected seven buttons. Each button corresponds to a different command/order for the robot.

Within updateButtonTab() method, we are constantly listening in a loop for a user input (click of a button).

When a button is pressed, we are writing its value to channel A, B, C and in the next step we are reading the value from the input COM OUT/IN into our button table (it represents the user input in the correct order), because the state of A ,B, C channel determines the value on the COM OUT/IN pin. The array is later interpreted to its corresponding order and stored in EEPROM.

Multiplexer PIN	Arduino PIN
A	9
B	13
C	8
COME OUT/IN	A5
VDD	5V
INH	GND
VEE	GND
VSS	GND

description of pins connection

Channel IN/OUT	Corresponding Command
0	Drive Forward
1	Drive Backwards
2	Turn Left
3	Turn Right
4	Start
5	Reset (Stop)
6	Wait (Pause)

description of buttons connection

3. EEPROM, a port of the Arduino’s processor

   A small hard disk of a robot, which total memory capacity is equal to 1024bytes.
   We are using it, to store an array of a user input. The process of writing to EEPROM, is executed in addOrderToTab() function. We are storing there respectively, the total number of orders and then the orders as an Integer values. The process of reading from EEPROM is executed in the main loop of the program and then passed as parameter (output) for the function which is responsible to commit orders, commitOrders(output).
   We have limited the user, so as he cannot put more than 10 commands at once, so even the EEPROM storage allows to store more commands, we forbidden that manually.

4. Shift Register SN74HC595N

   A register storage, constructed from a series of flip flops. It has been created to store a multiple bits of information, usually a binary data. We are using it control states of eight LEDs, which make up a user friendly interface. Every time a button is pressed, its corresponding LED blinks for 1 second. Beside that also during the command execution, the corresponding LED lights up. We have 7 LEDs for 7 commands and 1 LED (green) which is light up during code execution (Active Mode).

Function	Shift Register PIN	Arduino PIN
POWER	VCC	5V
GROUND	OE, GND	GND
DATA*	DS or SER	10
CLOCK**	SH_CP or SRCLK	5
LATCH***	ST_CP or RCLK	7
CLEAR****	MR or SRCLR	4

description of pins connection

* It determines which data is pushed to the register. It can store only digital values

**While there is a following state transition (LOW -> HIGH -> LOW), it adds a bit to the register

***While there is a following state transition (LOW -> HIGH -> LOW), it commits the shift register to the storage register

****While there is a following state transition (LOW -> HIGH -> LOW), it clears the shift register

Channel OUTPUT	LED of Corresponding Command
QA	Drive Forward
QB	Drive Backwards
QC	Turn Left
QD	Turn Right
QE	Start
QF	Reset (Stop)
QG	Wait (Pause)

description of LED connection

5. Tachometers and Photointerrupters TCST2103

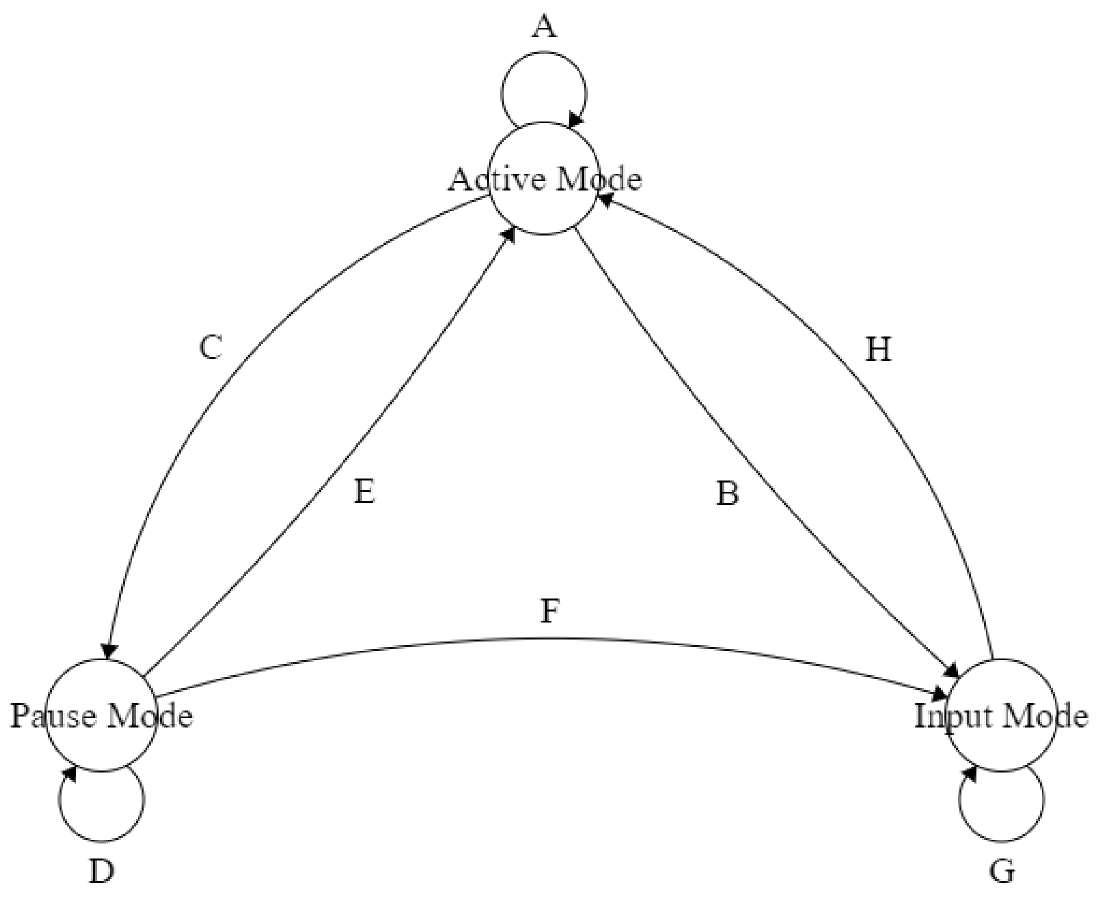
   A photointerrupter TCST2103 consists of a infrared emitter and a phototransistor, usually it is used to detect the interruption of the light flow. It is possible if the tachometer is attached between an emitter and phototransistor. A tachometer is a cog with a shrink edge. Everytime when it covers the emmiter, the light beam is interrupted.
   We are using the hardware (tachometers and photointerrupters) to precisely detect the right angle during a turn. No matter what is the speed value for the motors, the robot always turns 90 degree angle. It is done by using interrupted functions, which are triggered when its state is equal to CHANGED. The functions are just counters for a motor A and motor B.
   If we determine the precise number for the counters during turning 90 degrees, we can adjust the time for a turn automatically, basing on the input speed to preserve constant value for the approximated value of counters.

Interrupted PIN	Motor
2	A
3	B

description of pins connection

The robot finite state machine:

Our robot can be only in several states: Input mode, Active Mode and Pause Mode.



Description of a graph:

1. Active Mode:

A -> It is when the robots executes the commands from the EEPROM.

Transitions:

B -> When the “stop” button is pressed, the mode is changing to the Input Mode, so the user can put a new input or press “start” and execute previous orders from EEPROM. When the robot will execute all of its saved commands, the mode is changing automatically to the Input Mode

C -> When the “pause” is pressed (for a little longer) the mode is changing to the Pause Mode

2. Pause Mode:

D -> The robot is in a pause until the start is pressed

Transitions:

E -> When the "start" is pressed, the mode is changing to the Active Mode

F -> When the "stop" is pressed, the mode is changing to the Input Mode

3. Input mode:

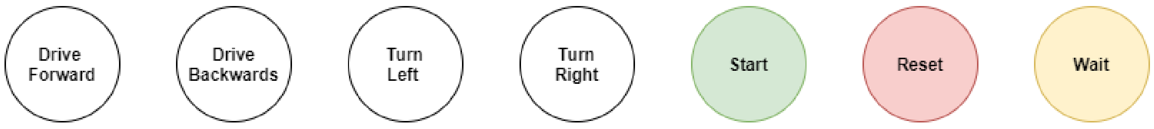
G -> It is when the robot registers user input (drive forward, drive backward, turn left, turn right, pause(as a delay).

Transitions:

H -> When the "start" button is pressed, the mode is changing to an Active Mode.

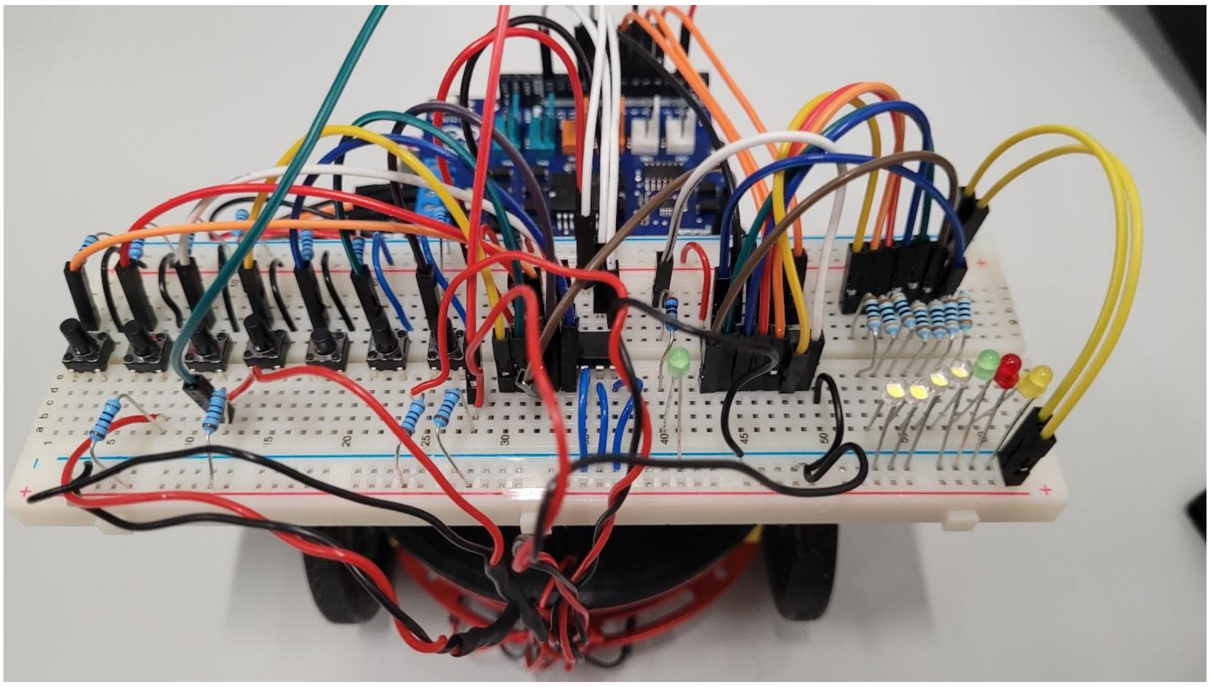
Our implementation:

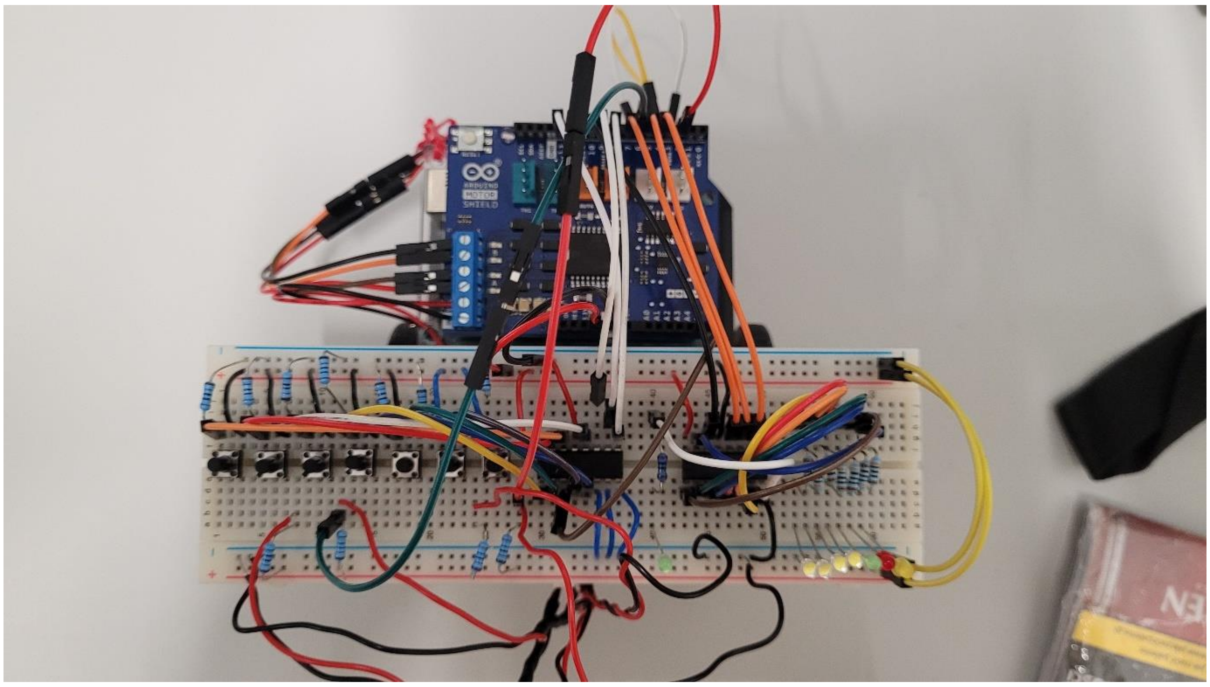
In our implementation, we can add up to 10 orders ahead. User have 7 buttons to play with and 8 LEDs each corresponding to given behavior. The buttons are arranged in one line and the illustration below shows their function and LEDs corresponding color:



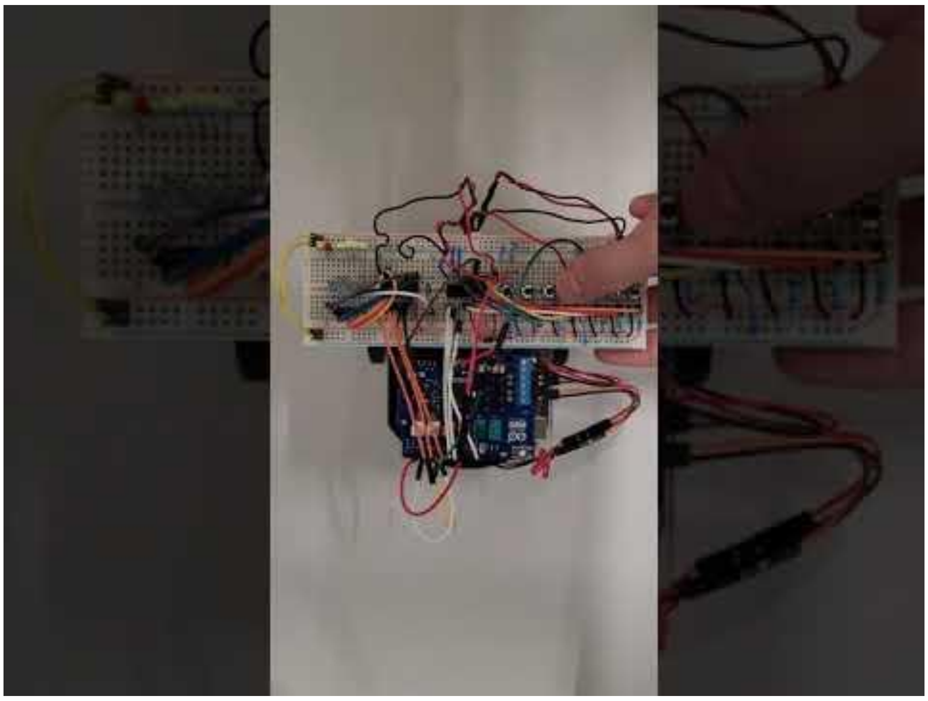
If there is space in orders buffer after pressing buttons Drive Forward/Backwards, Turn Left/Right or Wait the corresponding LED will light up for about a second. If buffer is full after pressing these buttons, LEDs will light up also but would not stay turned on after un pressing the button. Start and Reset buttons can be used any time. Reset clears the buffer and Start starts saved program. Wait button can be also used during execution of orders. It is used to pause robot work. To use this feature, user must press and hold wait button till a yellow LED will light up. Robot will stay paused till user press start button. After that, robot will resume his action at next order before pause. It is worth to say that all the commands are stored in the EEPROM memory, so after disconnecting the device from the power supply and reconnecting it, it will be able to execute the recently added orders. There is also one additional green LED which is turned on only while robot is currently following orders.

Photos of Robot:



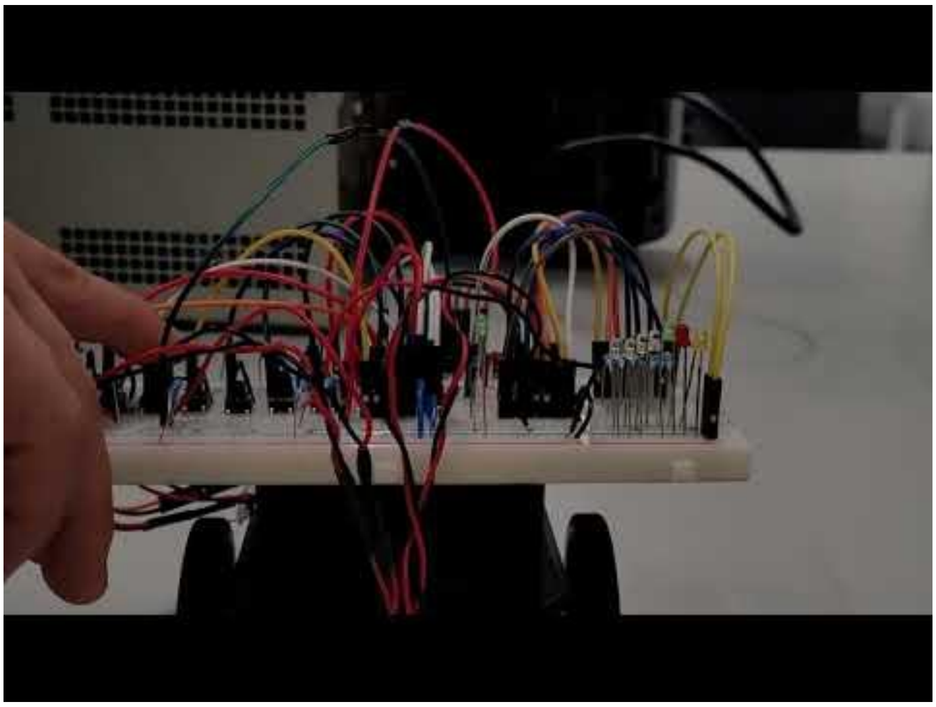


The demo below shows how the robot reacts when there is a space in EEPROM to take an order. Looking at LEDs behavior, you can determine that orders were read correctly. If the LED blinks once for a time of one second, it means that order was read correctly, otherwise not.



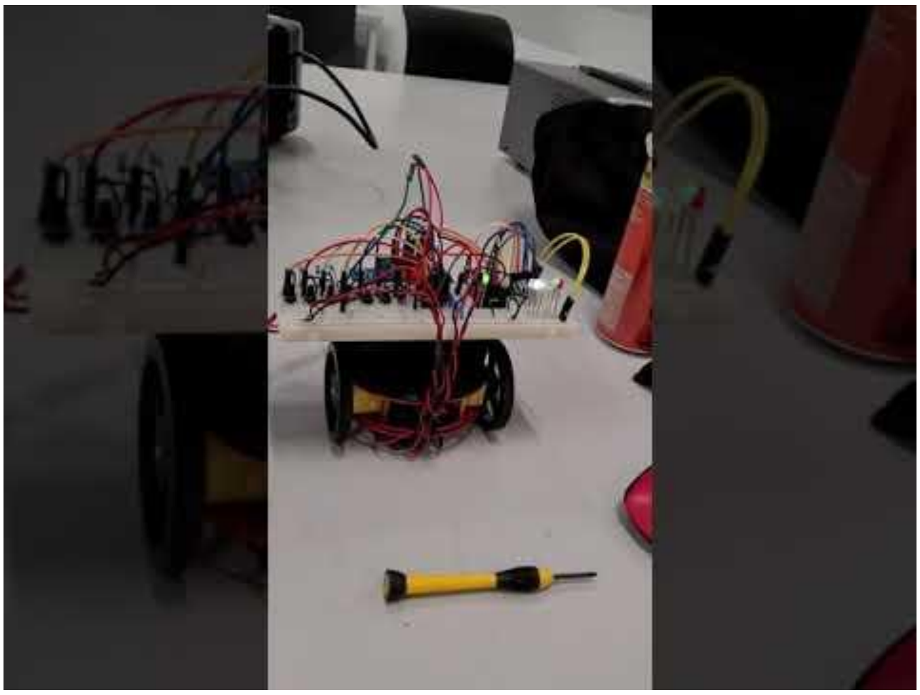
Buttons Demo

This demo shows behavior when the memory of the robot is full. It means that LDSs will be only lighting when the button is pressed, but any order wouldn't be added. After pressing start button, previously added orders are executed.



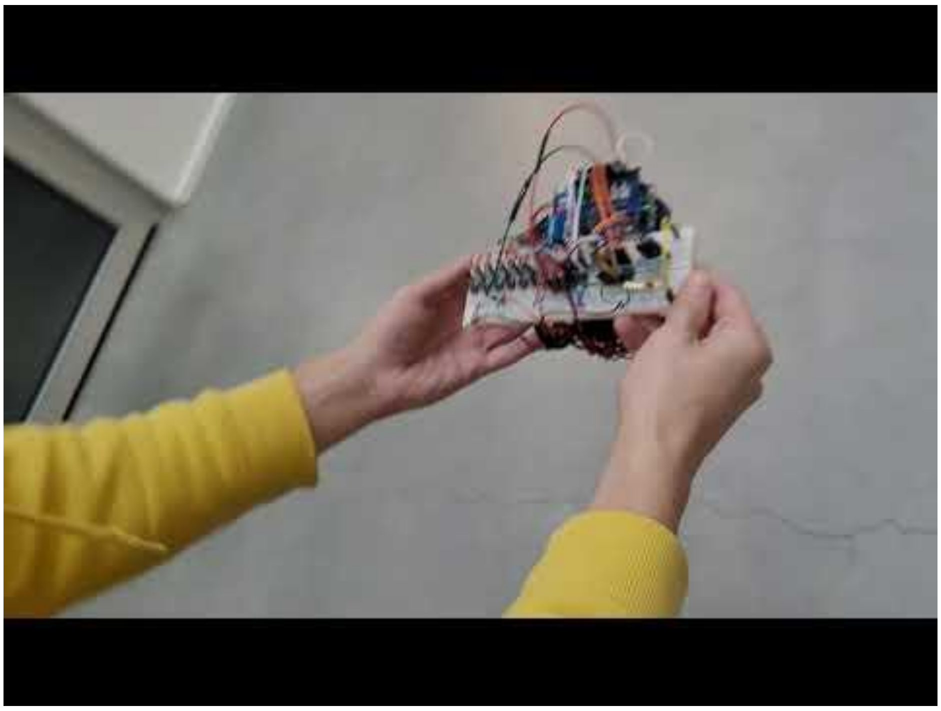
Full Buffor Demo

This demo shows that robot remembers last added orders. At the beginning we are adding few simple orders to the robot and executing it. Next, RESET button on Arduino MOTOR SHIELD is pressed, what is simulating switching power off and turning it on right after. Then the program is executed again with the same results as before.



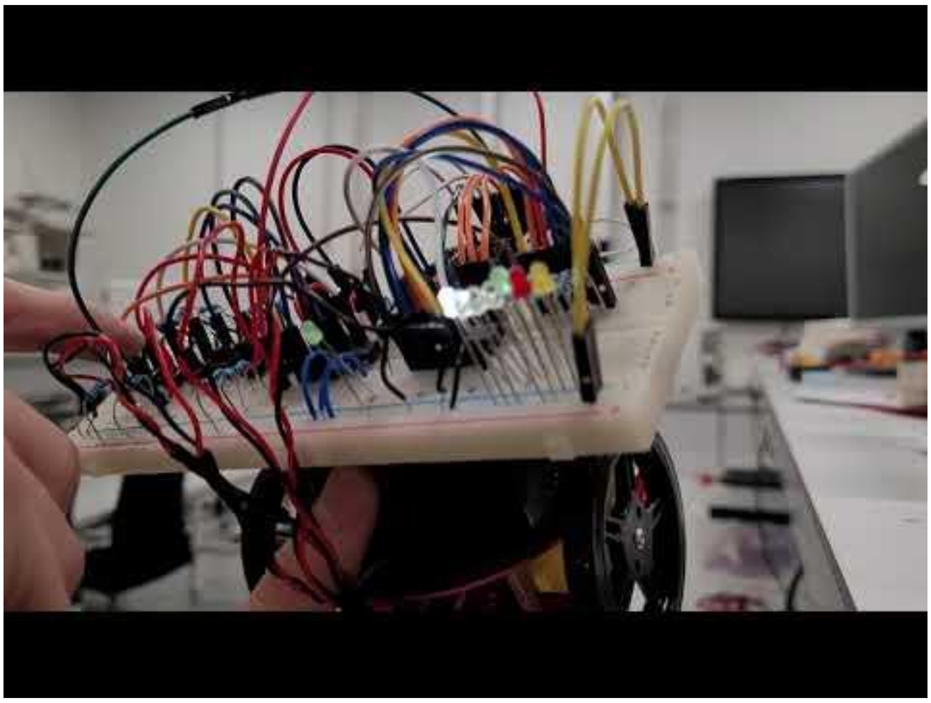
Saving last added orders in EEPROM

Last video in general demonstrate all functions available in robot.



Driving Demo

This demo shows how the reset, wait and pause feature works. At the beginning a sequence of orders are added: up, down, left, pause, right, left, down, up and then start button is pressed, so program starts executing orders (green LED on left side is turned on only when the program Is currently in active state). While first order is being executed, wait button is pressed and then the robot goes to paused mode and active state LED will turn off. It means that it will stay that till user will press start button. Later on, the pause and reset features are shown two more times.



[*wait & reset & pause demo*]

Source Code can be found at : *"Bee_Bot_final.ino"*