life.augmented

Introduzione alla robotica e al controllo motori con STM32

Arcangelo R Bruna

SRA AI SW & Tools
STMicroelectronics

life.augmented

Will-E description

Will-e overview agenda

1. Will-e: the goal
2. Hardware description
3. Firmware description
4. What’s next

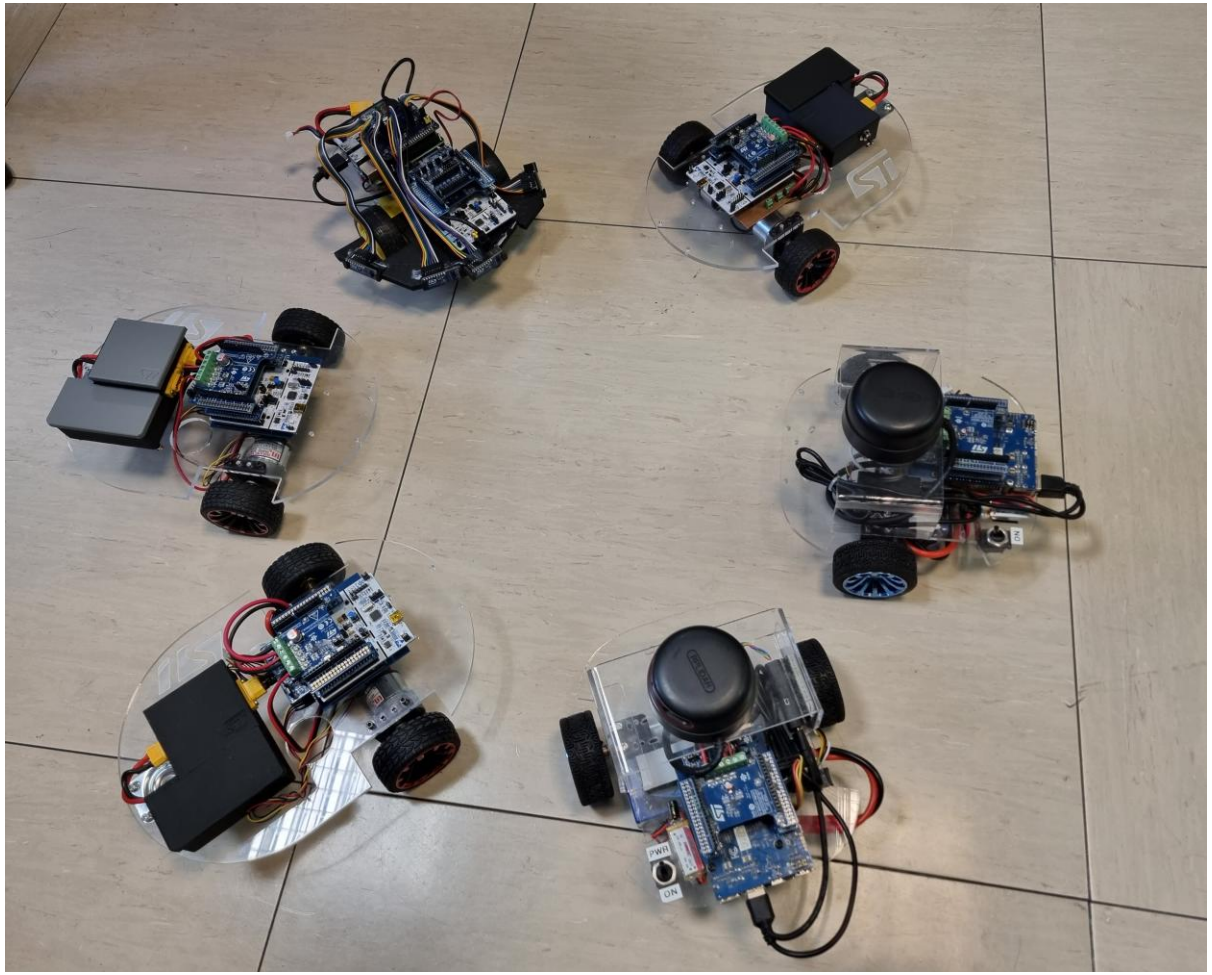
life.augmented

Will-e: the goal

life.augmented

Will-e: the goal

- The purpose of Will-e is to provide a basis for a robotic system.
- It is composed by
  - A physical frame with electric components (motors)
  - Electronics devices (microcontrollers, actuators, mems, power delivery)
  - Firmware to detects commands from different sources (e.g., serial comm, Bluetooth, etc.) and to execute basic commands (start, stop, go_straight, etc.)
- Additional boards allows to attach different devices for enhanced features
- Boards already tested:
  - STM32MP1 board
  - Raspberry board
- Additional sensors already tested
  - Imaging sensors (visible, Near Infrared, Short wave Infrared)
  - Depth sensors
  - IMU
- Additional communication methods tested
  - WiFi

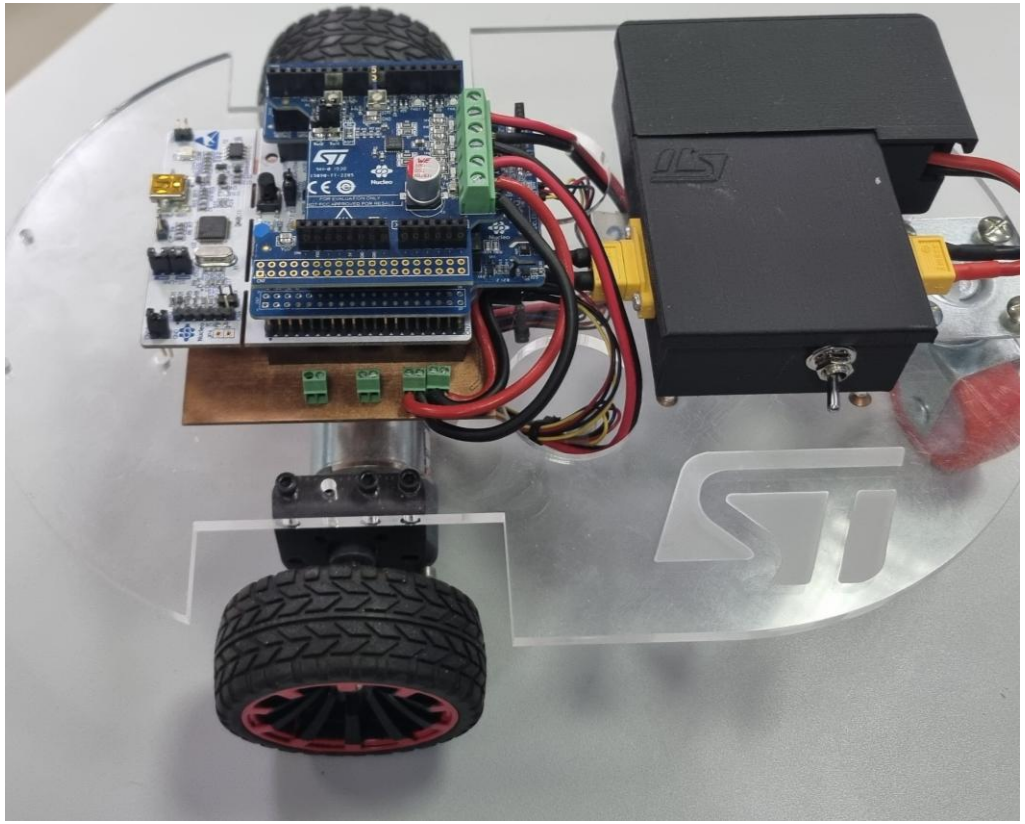


life.augmented

Hardware description

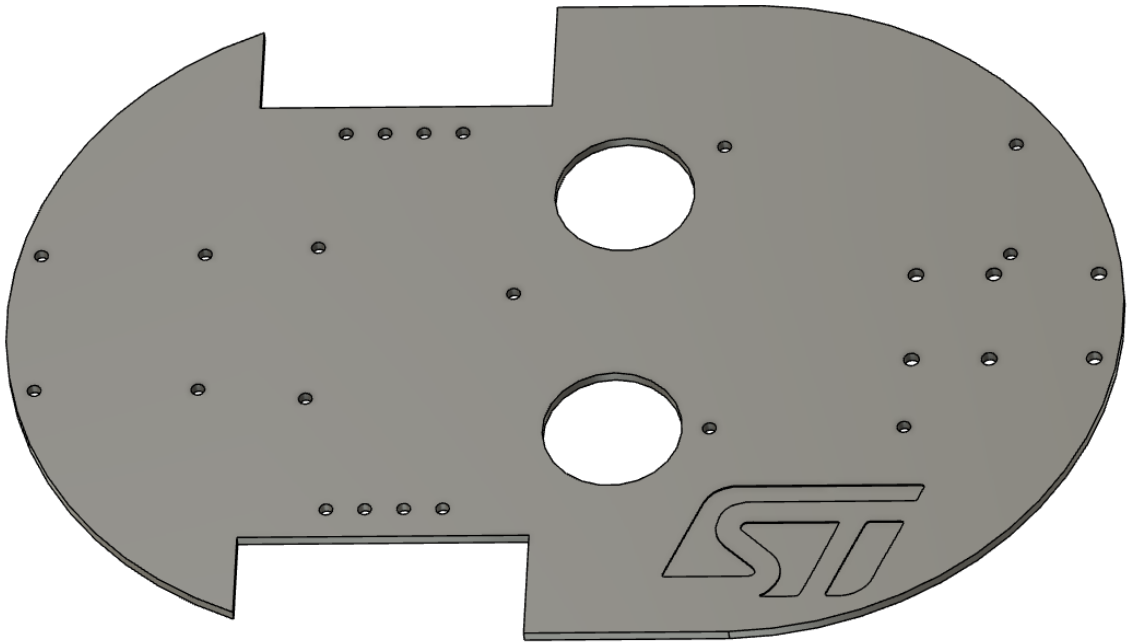
Will-e, Hardware Components

- The Will-e has been designed to have very simple, low-cost, and easily replicable hardware. It consists of:
  - Plexiglass body
  - 2 brushed DC motors and a passive wheel
  - Power: Li-Po battery, 3 cells, 11.1volts, 1300mAh
  - DC-DC step down converter, Vin: 11.1v, Vout: 7.5v
  - Main board: nucleo stm32f401
  - Expansion boards:
    - **x-nucleo-ihm15a1** Dual brush DC motor driver expansion board
    - **x-nucleo-iks01a3** Motion MEMS and environmental sensor expansion board
    - Custom designed expansion board



Plexiglass body

- Material: plexiglass
- Includes all the hardware devices
- Expandable: it allows to include host board, tof sensor board, etc



Motors, Wheels and Encoders

- 2 brushed dc motors:
  - HN-GH7.2-2414T model
  - Power supply: 7.5v DC
  - Gear ratio: 30:1
  - RPM: 291
- 2 Rubber wheels:
  - Diameter: 6.5cm
- Quadrature Rotary encoder:
  - 9000 pulses per revolution
  - Power supply: 5v

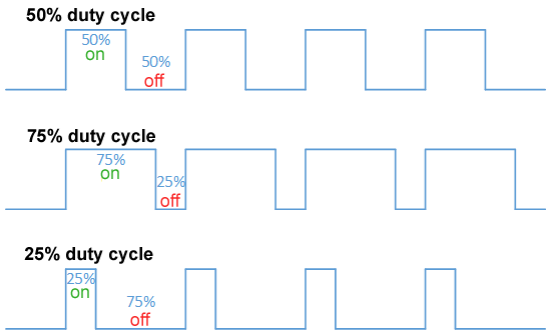
life.augmented

How to drive motors: PWM signal

- Using a variable DC source
  - Difficult to handle using a microcontroller
  - DAC + analog front end needed

Figure 1: Driving motor using variable DC source

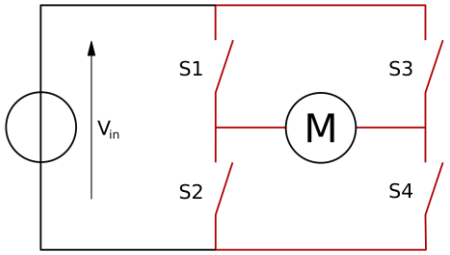
- Using a Pulse Width Modulation (PWM) signal
  - PWM is a signal shaped as a rectangular wave with a varying duty cycle
  - Simple with microcontrollers (with PWM peripherals)
  - It allows to modify the motor speed according to the duty cycle
  - Simple power amplifier is needed to drive motor

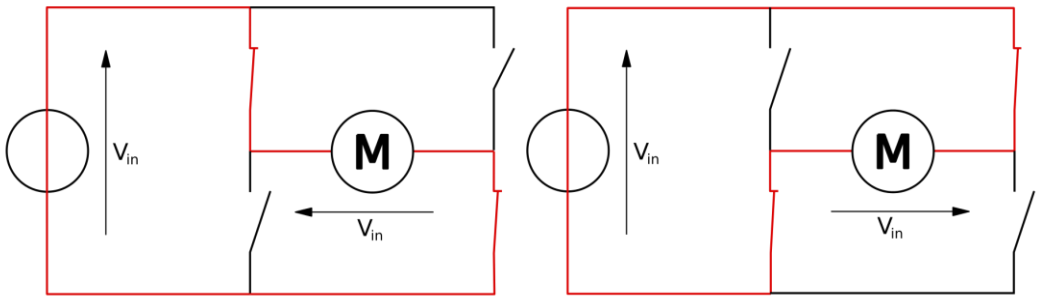


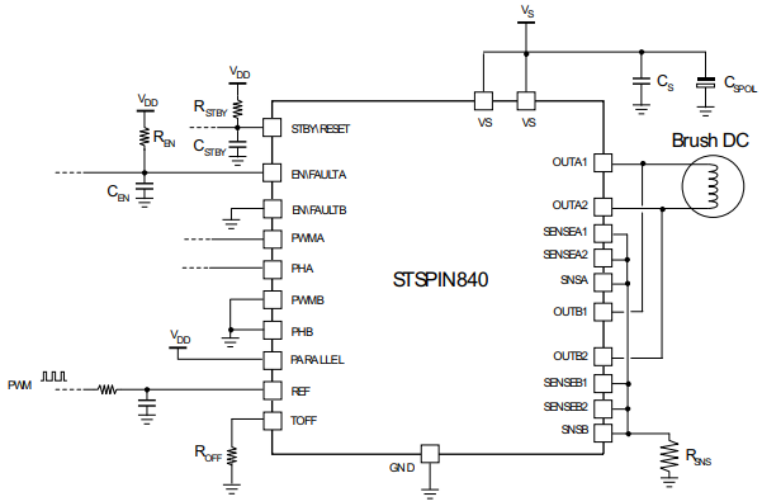
life.augmented

How to drive motors: H bridge

- An H-bridge is an electronic circuit that switches the polarity of a voltage applied to a load
- It is composed by (electronic) 4 switches
- Used in a motor, it allows changing the rotation direction
- STSPIN840 chip contains a complete H bridge and amplifiers with few external components



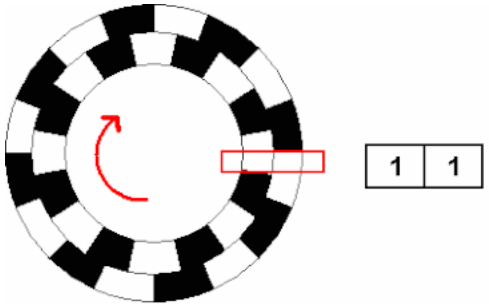


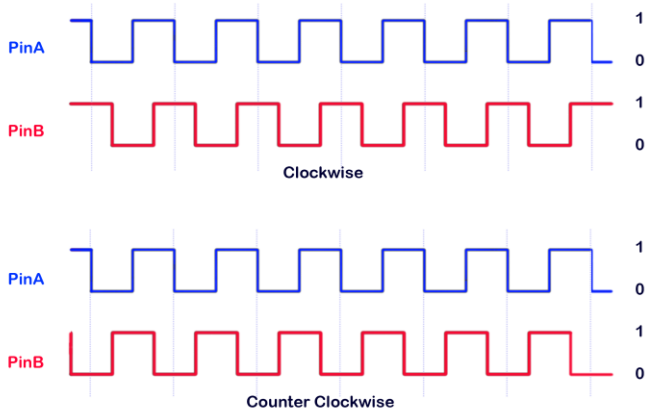


life.augmented

Rotary encoder

- An incremental encoder is a linear or rotary electromechanical device that has two output signals, A and B, which issue pulses when the device is moved
- Together, the A and B signals indicate both the occurrence of and direction of movement
- It allows to provide a feedback to the microcontroller about real rotation
- The number of pulses provides rotation angle
- Wheel dimension allows to retrieve linear movement
- Note: the measure fails when slipping happens!

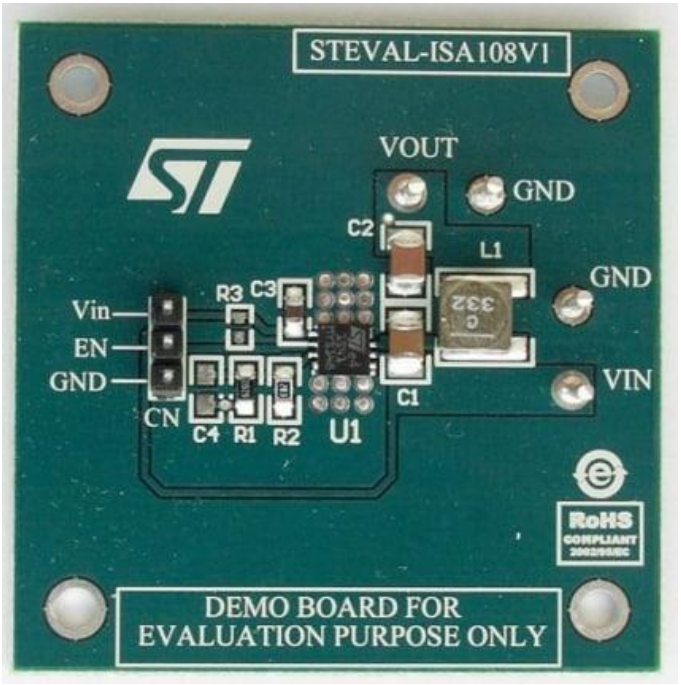




life.augmented

Power delivery

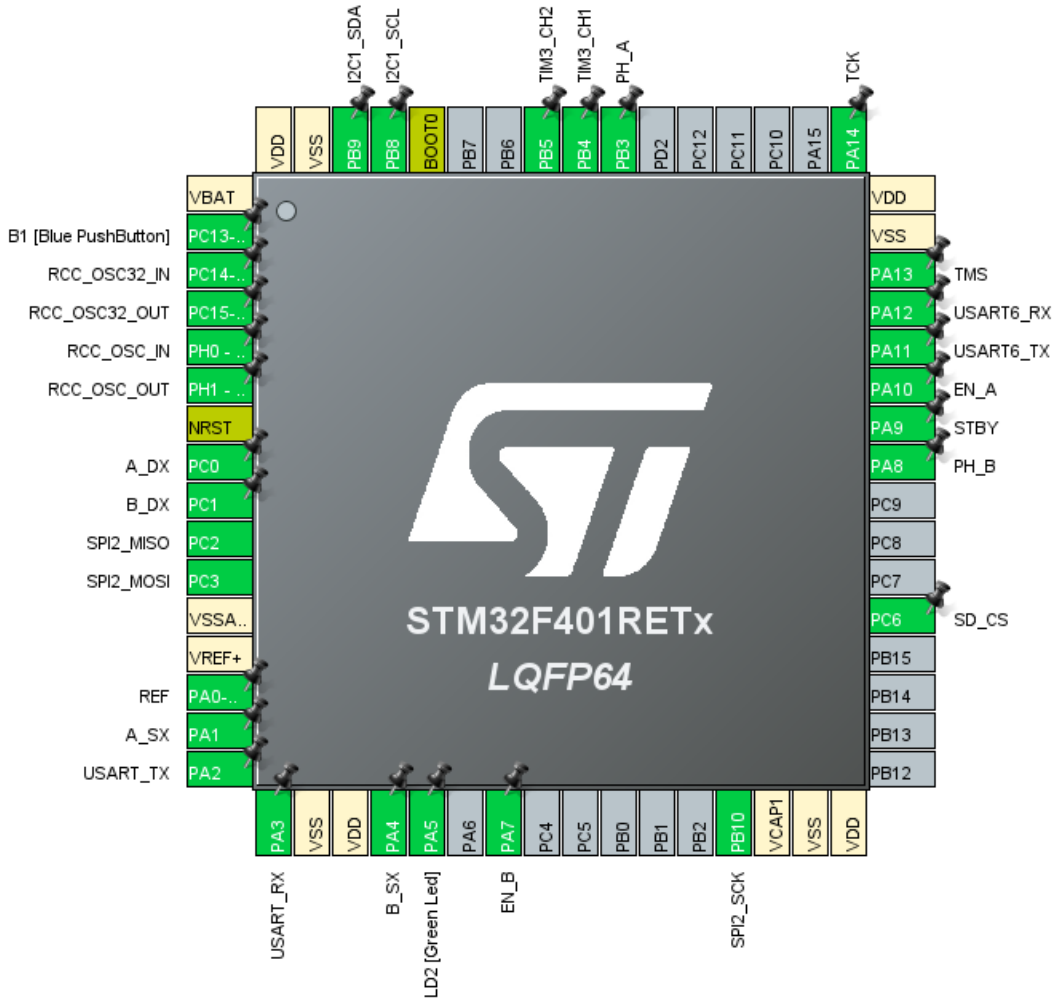
- Will-e is equipped with 11V battery
- Different parts needs different voltage, also depending on power needs
- DC-DC step down converters allows producing needed voltages
- STEVAL-ISA108V1:
  - 4 to 18 V input voltage
  - Output voltage adjustable from 0.8 V, 4A
  - 850 kHz switching frequency
  - Internal soft-start
  - Integrated 95 mΩ and 69 mΩ power MOSFETs
  - All ceramic capacitor
  - Enable function
  - Cycle-by-cycle current limiting
  - Current fold-back short-circuit protection
  - RoHS compliant



life.augmented

Main board

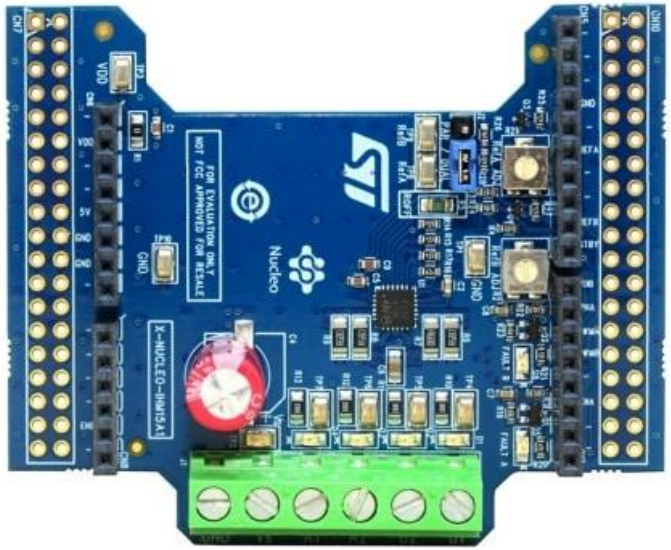
- nucleo STM32F401RE
  - 64 pin
  - Clock frequency: 84Mhz
  - Flash size: 512KB
  - RAM size: 96KB



life.augmented

Expansion boards

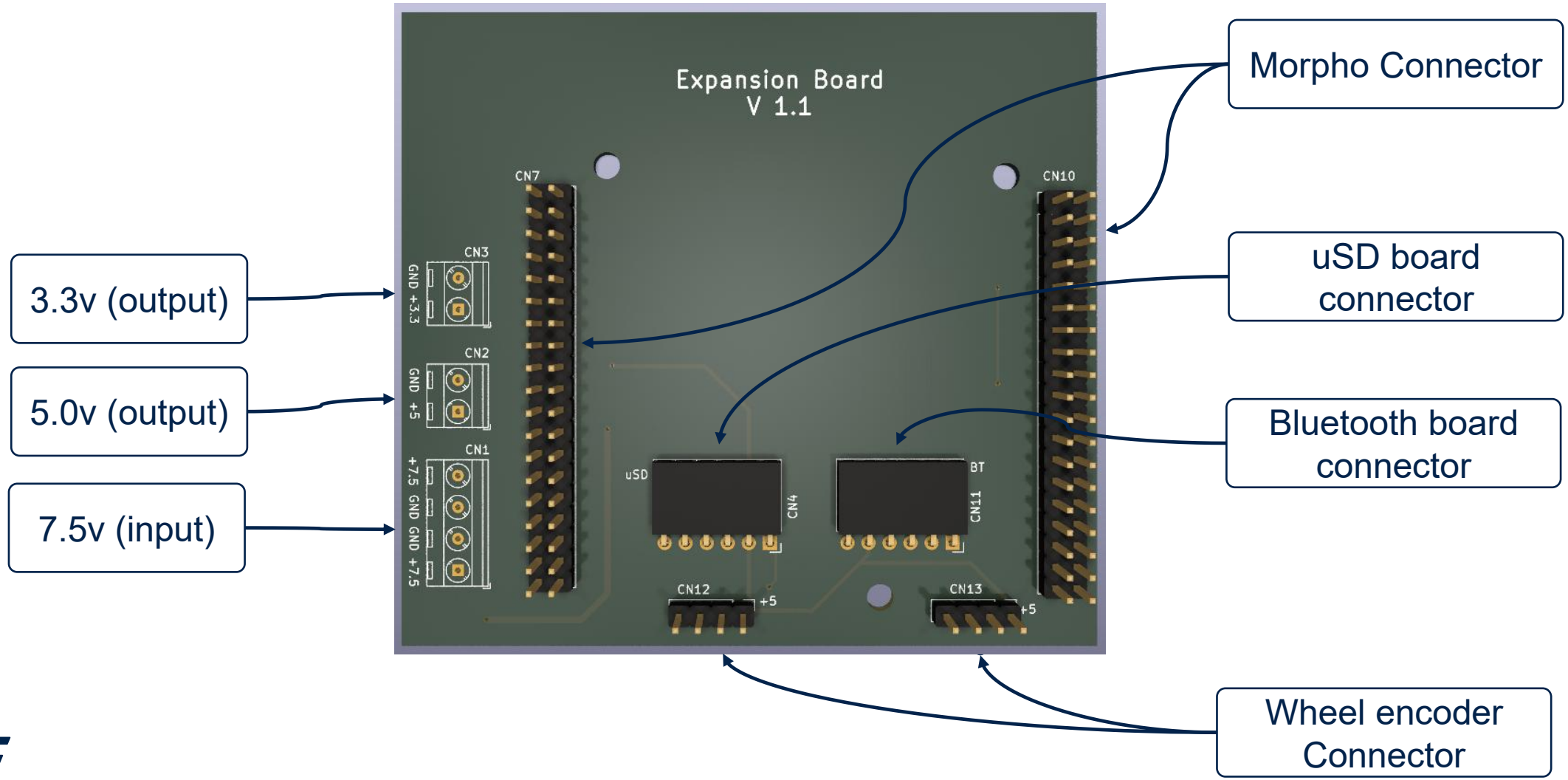
- x-nucleo-ihm15a1 Dual brush DC motor driver:
  - Voltage range from 7 to 45 V
  - Output current up to 1.5 Arms for each motor



- x-nucleo-iks01a3 Motion MEMS and environmental sensor:
  - LSM6DSO: MEMS 3D accelerometer + 3D gyroscope
  - LIS2MDL: MEMS 3D magnetometer +LIS2DW12: MEMS 3D accelerometer

life.augmented

Custom Expansion board



Firmware description

FW requirements

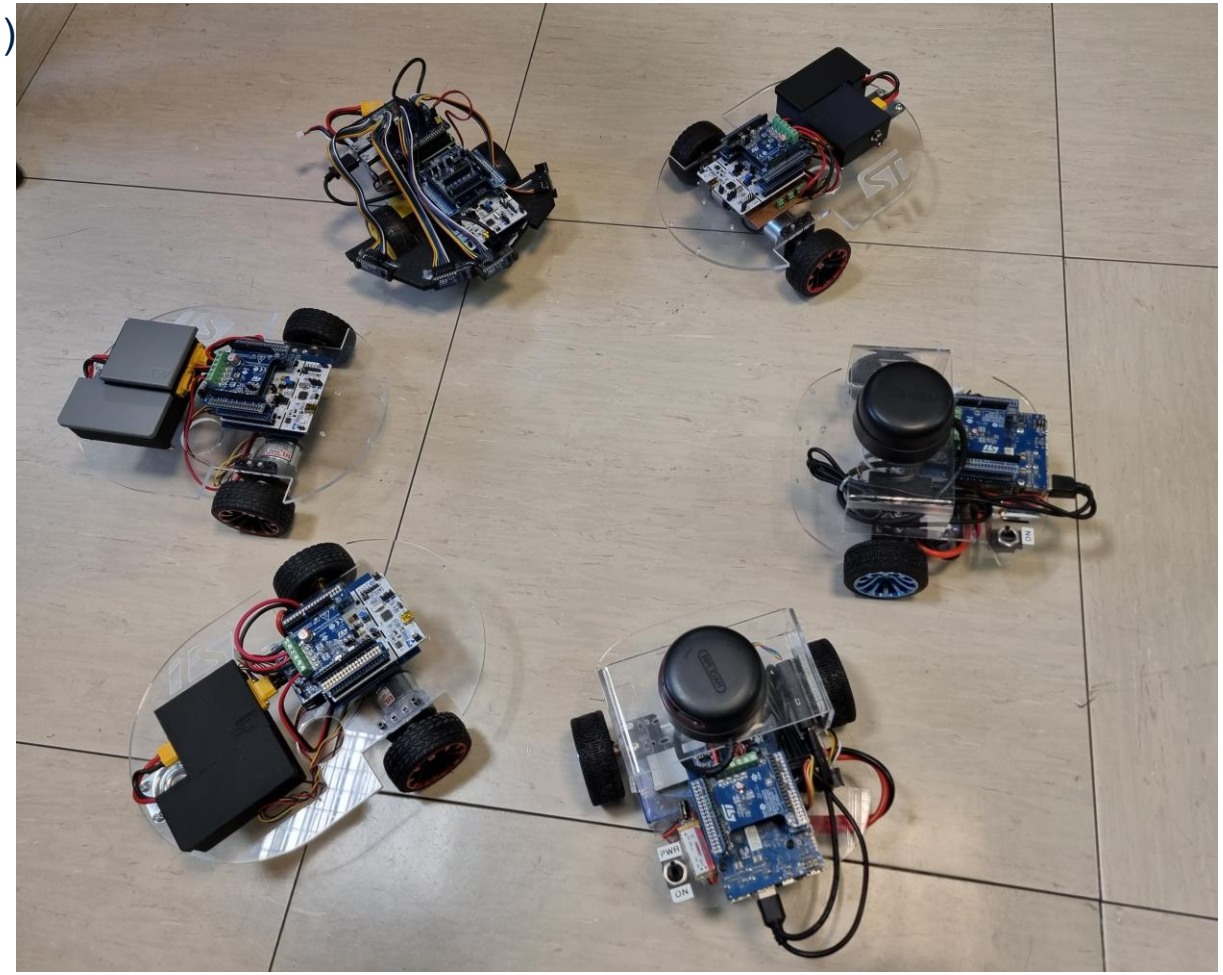
- The FW needs to:
  - Receive and decode commands (API based)
    - human or automatically generated
  - Execute commands
  - Estimate and Update Odometry coords (x, y, θ)
  - Save all data onto microsd (datalogger)
  - Stream Odometry coords to selected peripherals (e.g., for navigation purposes)

FW requirements

- The fw needs to:
  - Read uart host commands, human or automatically generated, and execute them
  - Go forward/backward, at a set cruise speed, covering a set distance
  - Rotate, or travel a curve
  - Estimate and Update Odometry coords (x, y, θ)
  - Save all data onto uSD
  - Stream Odometry coords over bluetooth

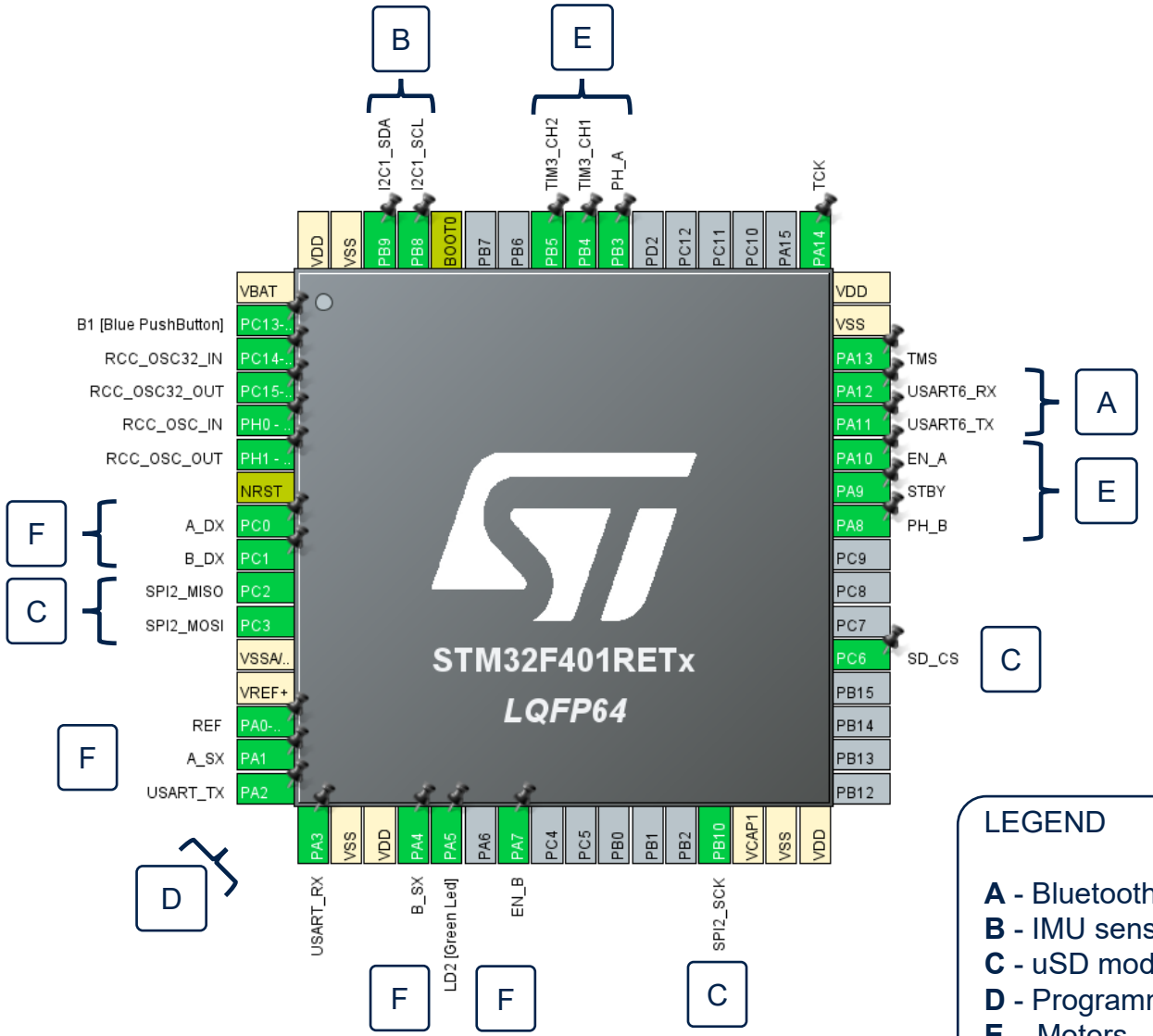
Basic Commands

- Commands are received by any peripheral (USART preferred)
- Commands are in plain text (for human readability)
- Several commands have been implemented:
  - Go forward/backward, indefinitely
  - Go forward/backward, *x* cm
  - Set cruise speed to *x* cm/s
  - Set rotation speed to *x* cm/s
  - In place rotation by *α* degrees
  - Travel a curve with a radius *R*, rotating by *α* degrees
  - Get and update odometry coords *(x, y, θ)*, dead reckoning
  - Get IMU values (accelerometer, gyroscope and magnetometer)
  - Start/Stop saving all available data into a uSD
  - Start/Stop streaming data over Uart peripheral



FW – Peripheral Settings

- Main clock frequency: 84Mhz
- Peripherals:
  - Usart2 (Programmer and PC)
  - Usart6 (Bluetooth module)
  - SPI2 (uSD module)
  - I2C1 (IMU module)
  - TIMER3 (PWM, motors)
- Interrupts:
  - From wheel encoders
  - From uarts
  - From button

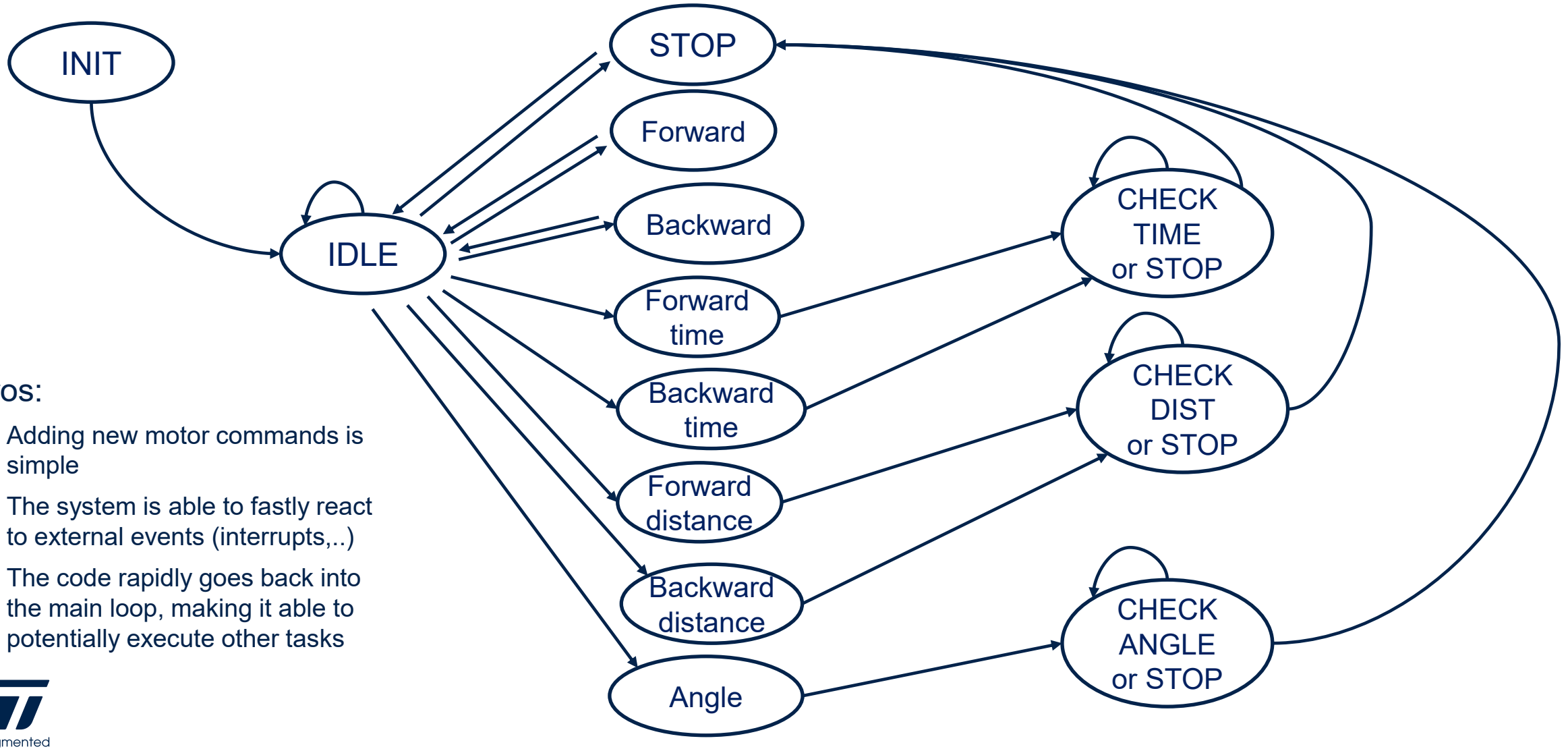


LEGEND

A - Bluetooth module
B - IMU sensors module
C - uSD module
D - Programmer uart (PC)
E – Motors
F - Encoders

life.augmented

FW: The FSM



- Pros:
  - Adding new motor commands is simple
  - The system is able to fastly react to external events (interrupts,..)
  - The code rapidly goes back into the main loop, making it able to potentially execute other tasks

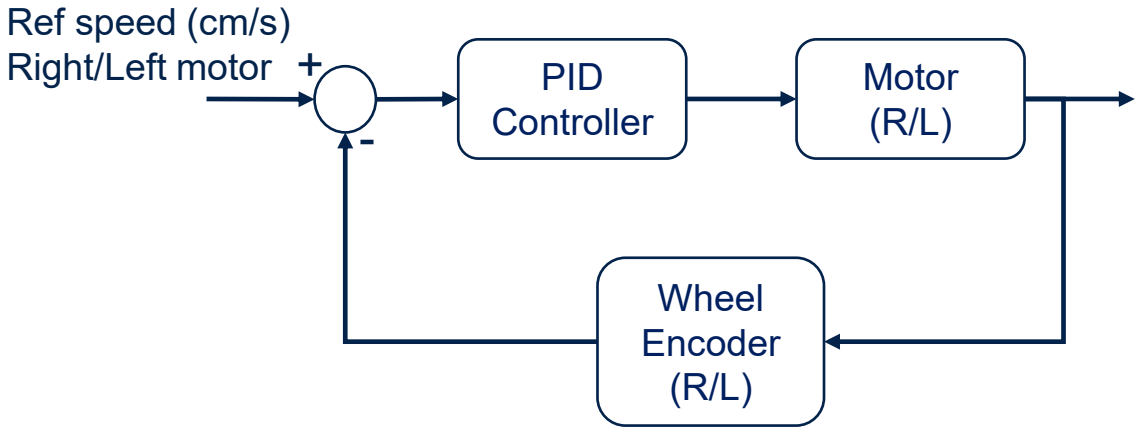
life.augmented

Ideal vs real case

- In ideal system, everything works as modeled
- In real system, tolerances provoke faults in formulas
- E.g.: the real wheel movement is affected by:
  - Real wheel size
  - PWM precision
  - Motor rotation precision
  - Inertia (usually more power is needed to the motor to start rotating)
- A feedback and a loop control is needed, e.g., to go straight !

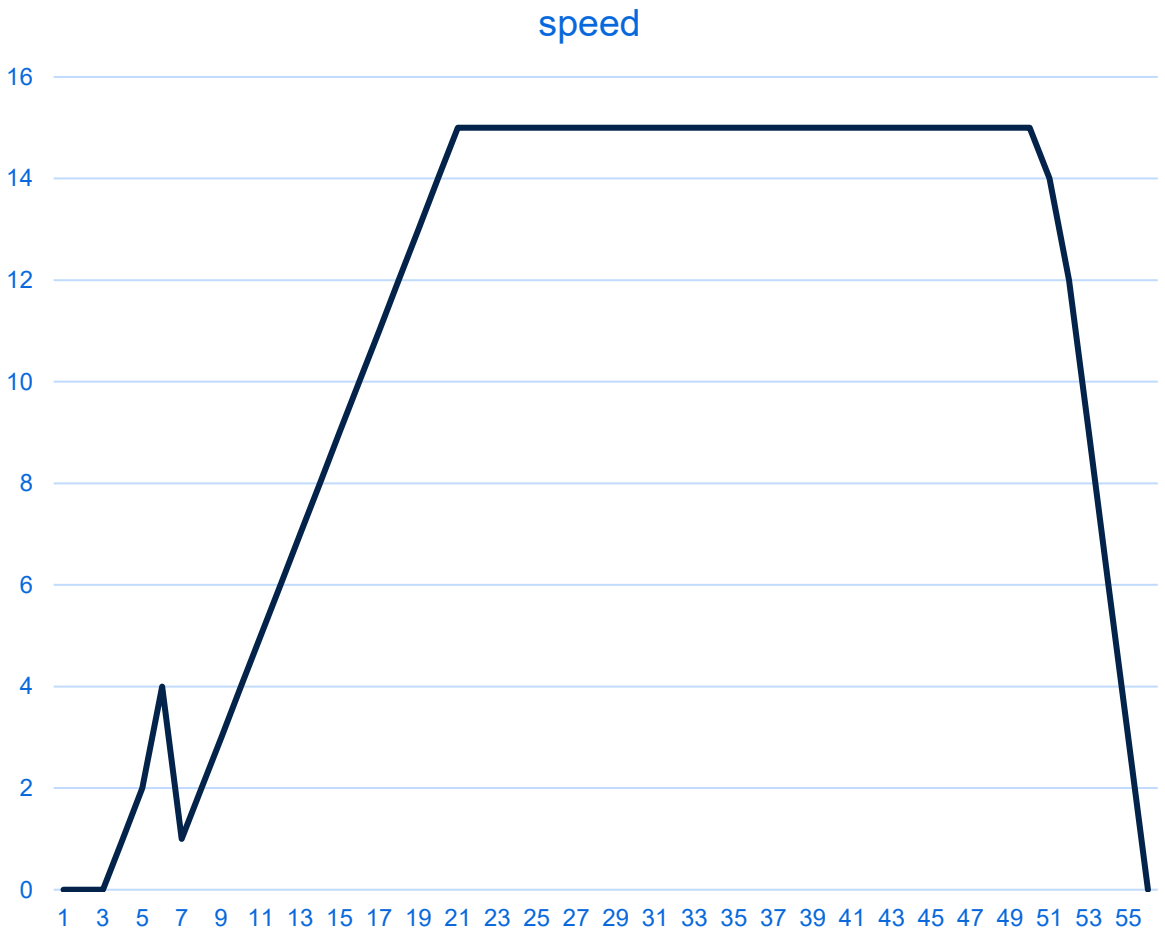
FW - challenges

- Without the Control Loop, the motor behaviour is unstable, resulting quite difficult:
  - To softly start
  - To go (really) straight
  - To reach and maintain the cruise speed
  - To stop at the right distance
  - To update odometry
  - To rotate through a set angle
  - Travel a curve with a radius R



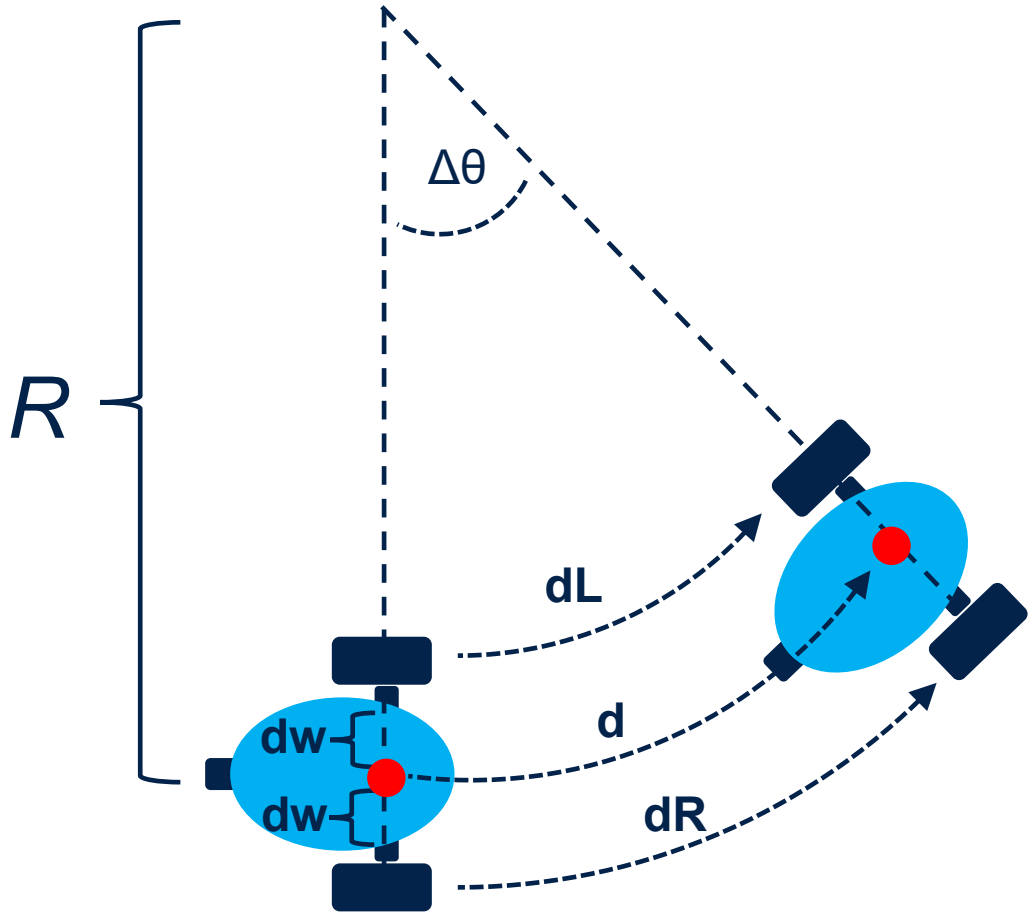
- In the control loop we use the wheel encoder sensor
- In the final version, the PWM output is set @100% to break motor inertia, then we go down to a low PWM values, and then a ramp on reference speed is applied, until reaching the user reference speed

FW – How to go straight



- To go straight, the two wheels must rotate at the same speed (the cruise speed, set by host)
- When the motor has to start:
  - The PWM is set to 100% to make the motor start
  - Immediately after each motor starts, the wheel encoder detects it, the PWM is then scaled to the minimum value that allows the wheels to rotate slowly
  - A cruise speed ramp is applied, until reaching the required host cruise speed. This behavior avoids sudden accelerations
  - When the stop command is reached, the PWM value is set to zero and the wheels stop rapidly

FW – Rotate



dL: distance traveled by the left wheel

dR: distance traveled by the right wheel

dw: distance between the reference point and the wheels

d: distance traveled by the reference point

Δθ: change in the angle of rotation

R: radius of the curve containing the reference point

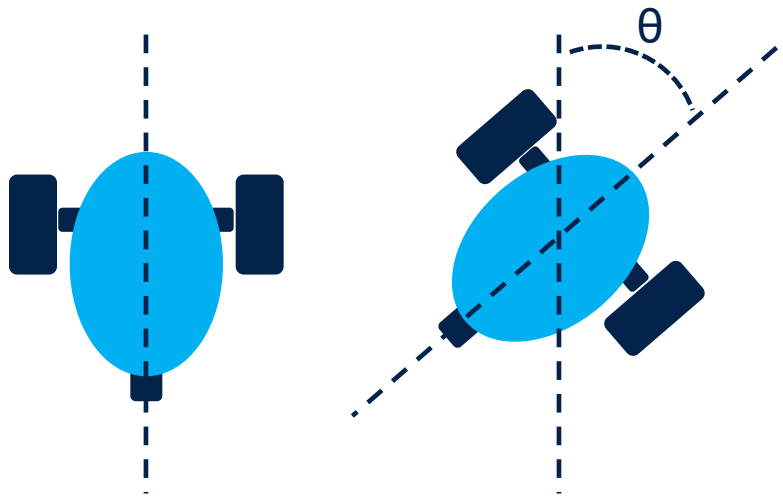
$$d = \frac{dR + dL}{2}$$

$$\Delta\theta = \frac{dR - dL}{2dw}$$

life.augmented

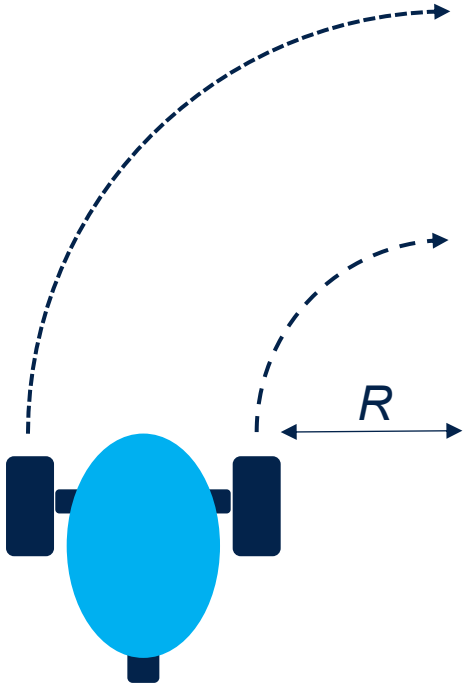
FW – in place rotation

- In order to rotate in place, the wheels must rotate in opposite directions, at the same speed
- Once the requested angle is reached, the wheels stop



FW – go around a curve

- To go around a curve, two different speed must be set, one for the inner wheel, one for the outer wheel
- A radius *R* must be set, too
- Once the requested angle is reached, the wheels stop



$$\begin{cases} v_i = \dfrac{2\pi R}{t} \\ v_o = \dfrac{2\pi(R + 2dw)}{t} \end{cases} \Rightarrow v_o = \frac{2\pi(R + 2dw)}{t}$$

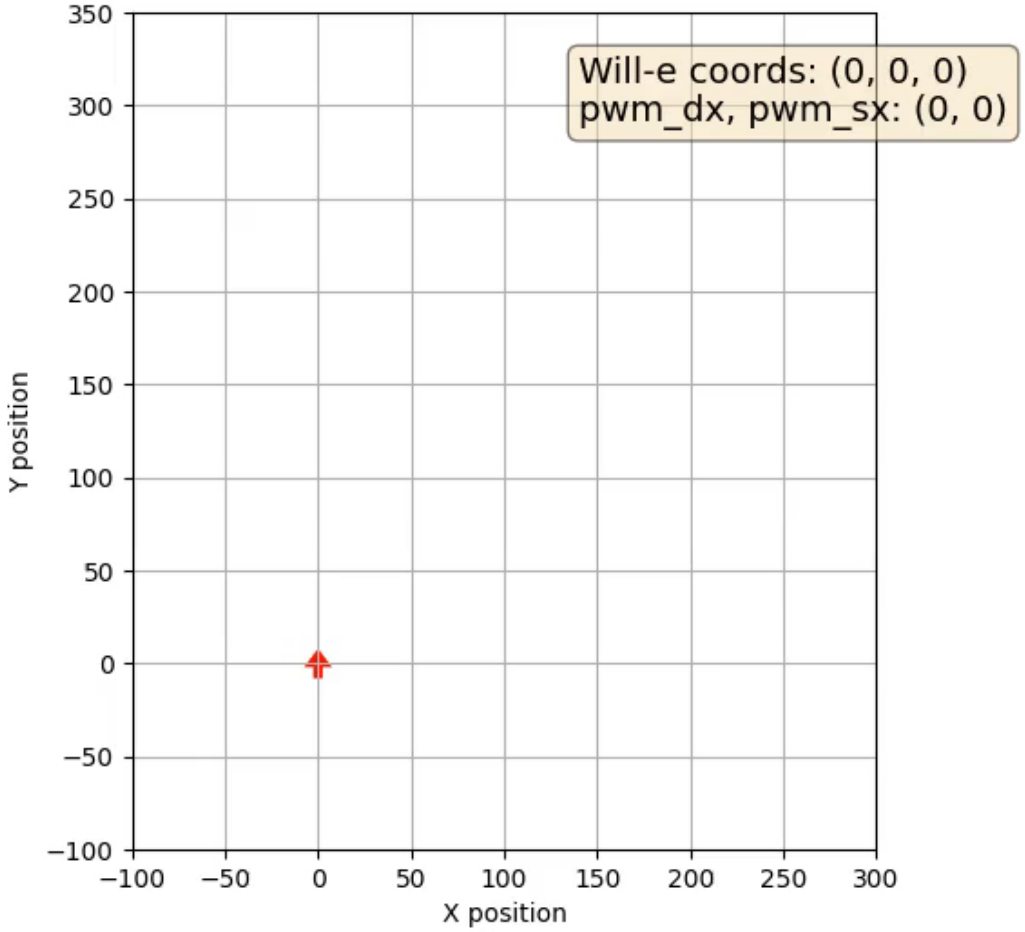
life.augmented

FW – save data into uSD

- New command: ***start_write_sd*** (and ***stop_write_sd***) allows to save all collected data into a uSD
- Data is saved @26Hz (IMU output data rate) in csv format
- Data can be used to develop new offline algorithms, navigation strategies, sensor merging, etc
- Collected data:
  - ***Timestamp:*** the microcontroller timestamp, in ms
  - ***delta_ticks_dx, delta_ticks_sx:*** ticks collected by the wheel encoder, from the previous streamed data
  - ***x, y, theta:*** updated odometry value
  - ***pwm_dx, pwm_sx:*** current pwm, for right and left motor
  - ***acc_x, acc_y, acc_z, gyro_x, gyro_y, gyro_z, magn_x, magn_y, magn_z:*** data from IMU

SW – plot saved data

- *A python script was developed to plot data, in offline mode*
  - *In the example, the executed commands are:*
    - *start_forward_distance 300*
    - *angle_control 90*
    - *start_forward_distance 60*
    - *angle_control 90*
    - *start_forward_distance 60*
    - *angle_control 90*
    - *start_forward_distance 60*
    - *angle_control -90*
    - *start_forward_distance 240*
    - *go_curve -270*



life.augmented

What next

What’s next

- Adding new commands:
  - ***enable_streaming*** and ***disable_streaming***, to stream data via bluetooth, in real time
- Under development a pc graphic user interface, in order to plot data, in real time
- Odometry algorithms integrating IMU sensor data?
- Converting all the fw to make it a microROS / ROS2 node?

Our technology starts with You

Find out more at www.st.com

life.augmented