life.augmented

Introduction to Robotics with STM32

Giuseppe Messina - AIST

Overview

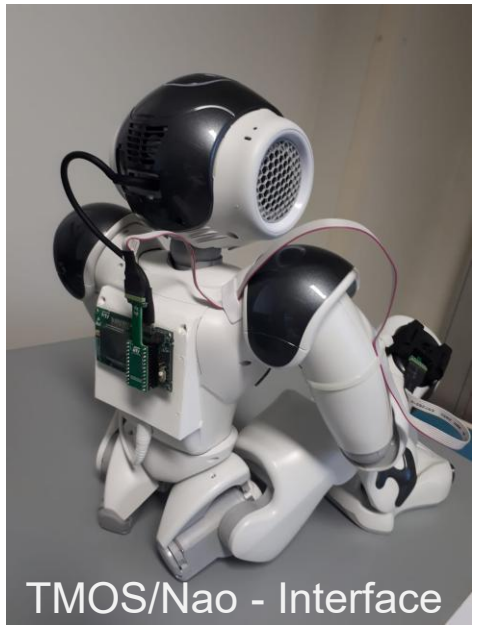
- The goal of ST's robotics activities is to incrementally integrate ST's technologies and algorithms developed and to be developed in robotic environments.
- Explore current scenarios and anticipate the future use of microelectronics in robotics.
- The current scenarios we are exploring concern
  - Indoor Navigation and Simultaneous Localization and Mapping (SLAM)
  - Human-robot interaction
  - Robotic Assistance - Cobot

life.augmented

Robot Nao e Tecnologie ST

- NAO
- Based on Softbank's Nao Robot, our project aims to integrate ST devices into the existing environment as much as possible;
- Built-in TMOS sensor for measuring human body temperature
- The first demo was presented at STMicroelectronics Technical Community Day 2019, in collaboration with AMS's R&D division, as part of the AMICO (National Funded Project for Post-Operative Robotic Assistance) project.


TMOS -Sensor

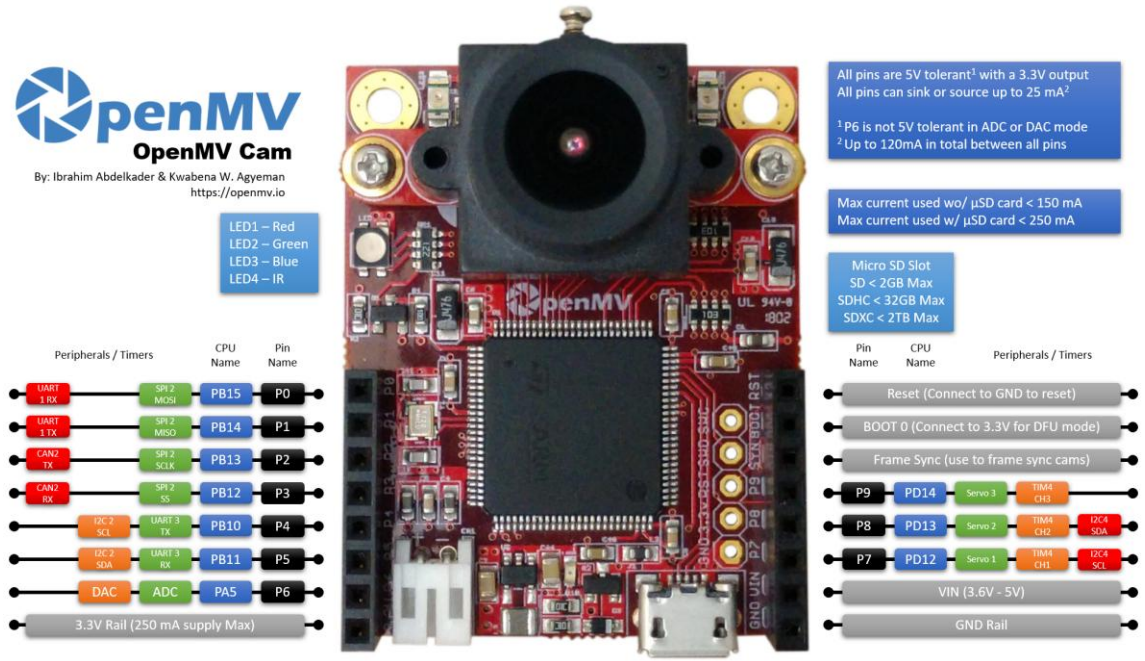

TMOS/Nao - Interface

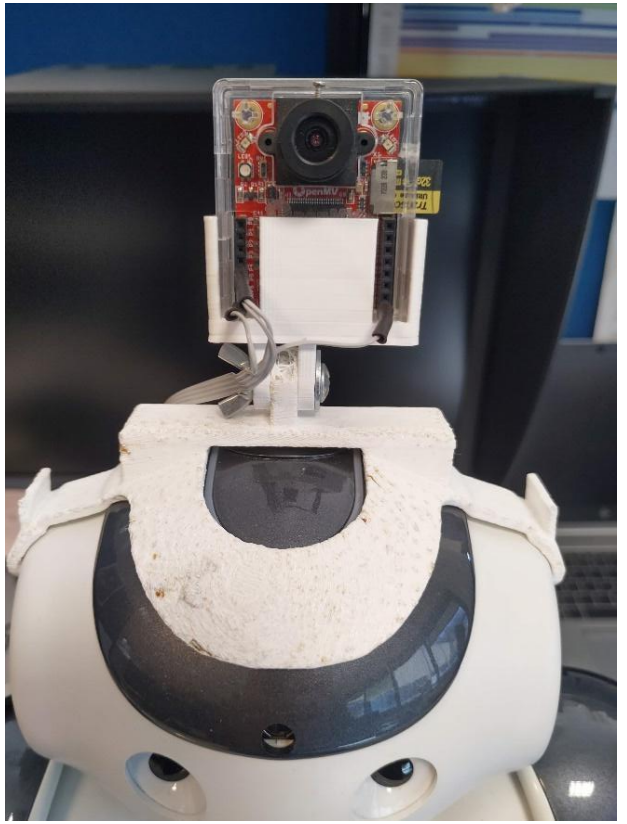
life.augmented

Nao Robot and ST tecnologies

Face recognition and Smile Detection to pilot Nao movements

- Using a ST firmware developed for the OpenMv camera based on the SMT32H7 microprocessor, a facial recognition and smile detection algorithms has been integrated.
- The OpenMv camera has been connected to Nao enable Face and Smile detection and to perform a reaction simulated by the robot.

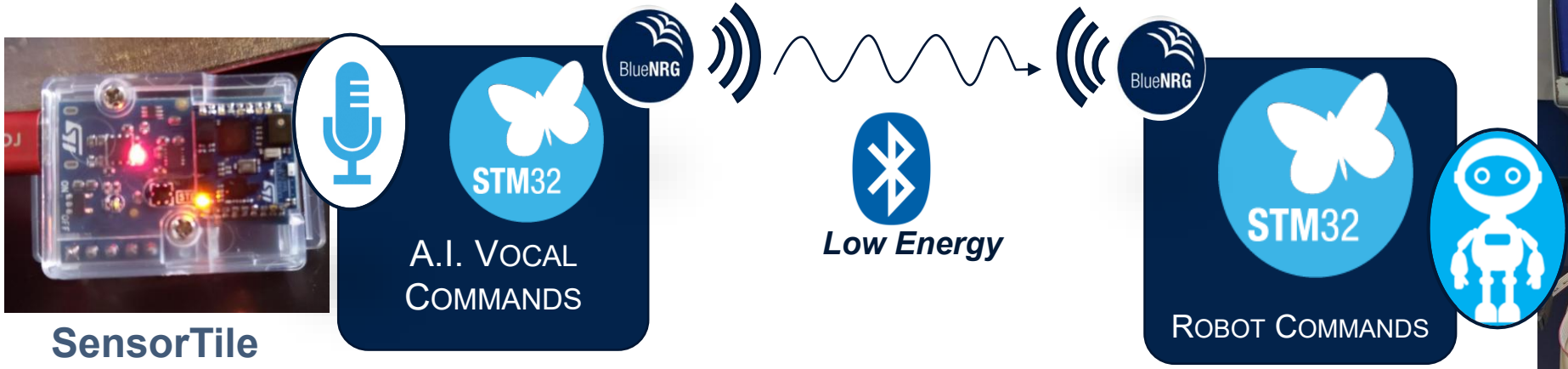




Authorized Partner

life.augmented

Robot Nao e Tecnologie ST



Voice command recognition to control Nao's movements

Two boards: **SensorTile and BlueCoin**

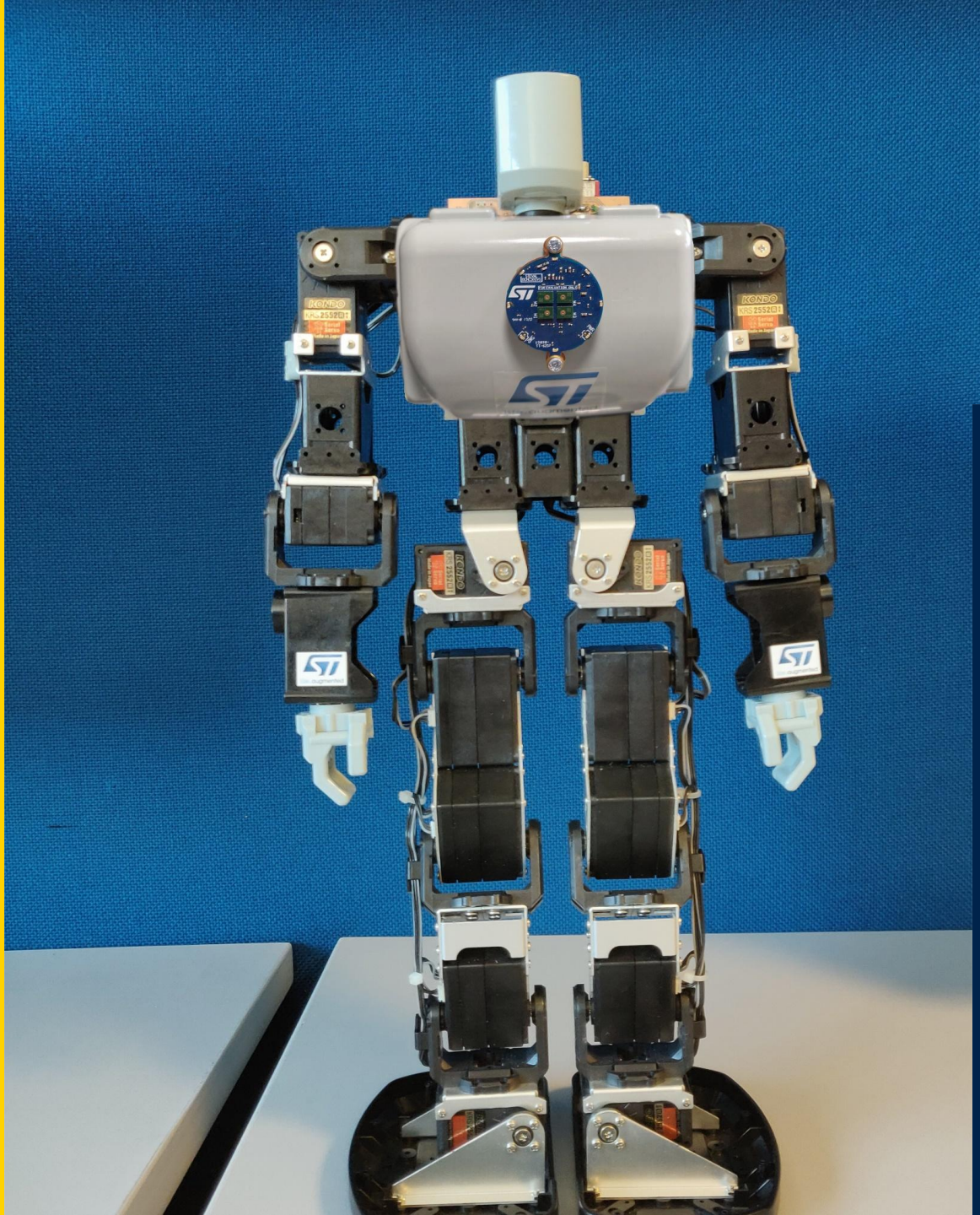
- The two boards communicate via **BTLE**
- The SensorTile contains an RNN trained to recognize **6 commands**
- Nao executes commands received via BTLE

life.augmented

Demo - video

AMICO project, admitted to MIUR funding, area of specialization "Technologies for Living Environments"

This work was presented at Maker Faire Rome 2020

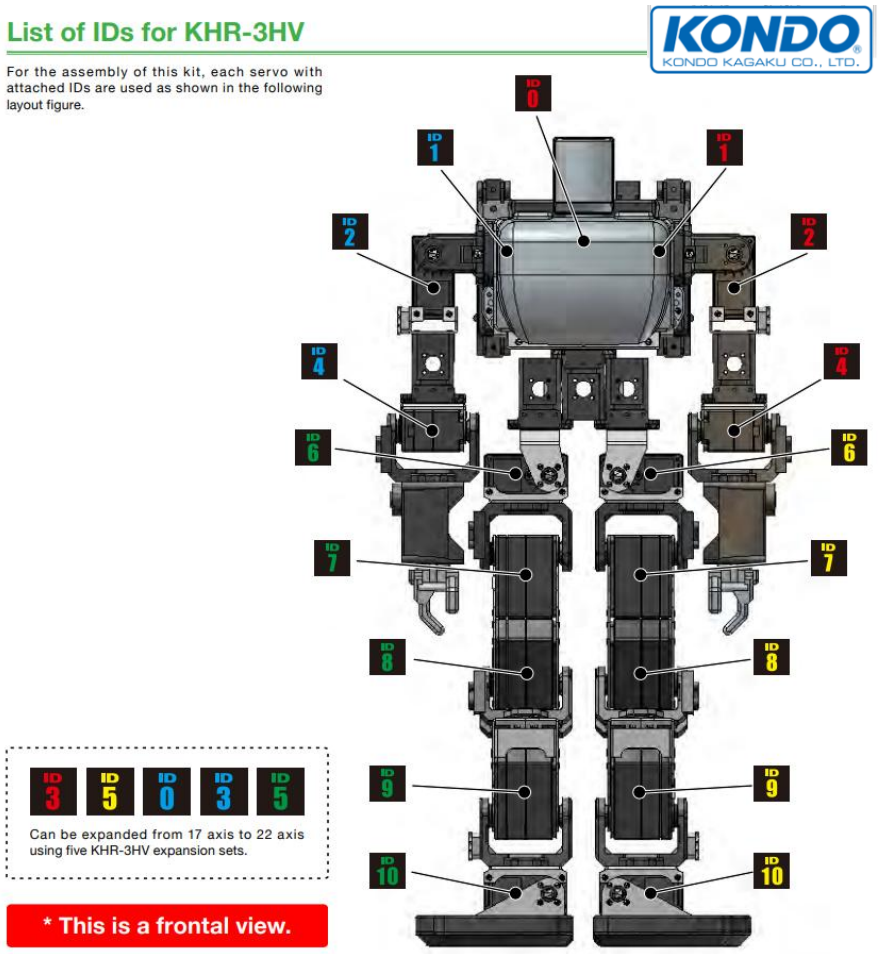


life.augmented

Humanoid Robot System

Original Kondo KHR-3HV

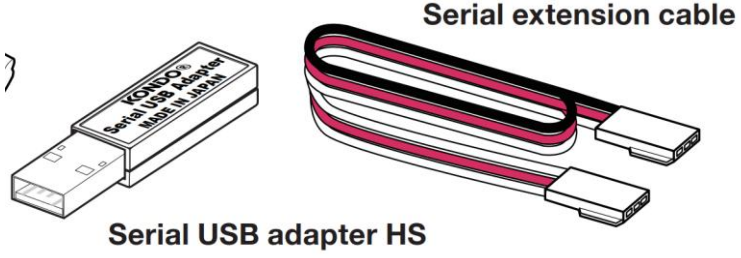
- Robot KONDO's KHR-3HV, made by the Japanese company Kondo Kagaku co.
  - 17 active servo motors and the predisposition for 5 more.
  - MCU **Renesas RCB-4HV**, capable of piloting up to 35 servo motors.
  - Compatible with the ICS 3.5 motor protocol
  - The servo motors used are the KRS-2552RHV. They can operate with the Daisy Chained protocol, allowing single-chain servo motors to be connected:
    - Maximum Operating Angle 270º
    - Maximum Holding Torque 14kgf.cm (11.1V)
    - Speed 0.14s/60º (11.1V, under no load)
    - ICS 3.5 protocol (Serial and PWM comunication)
    - Size 41x21x30.55 mm
    - Weight 41.5g
    - Operating Voltage 9V~12V

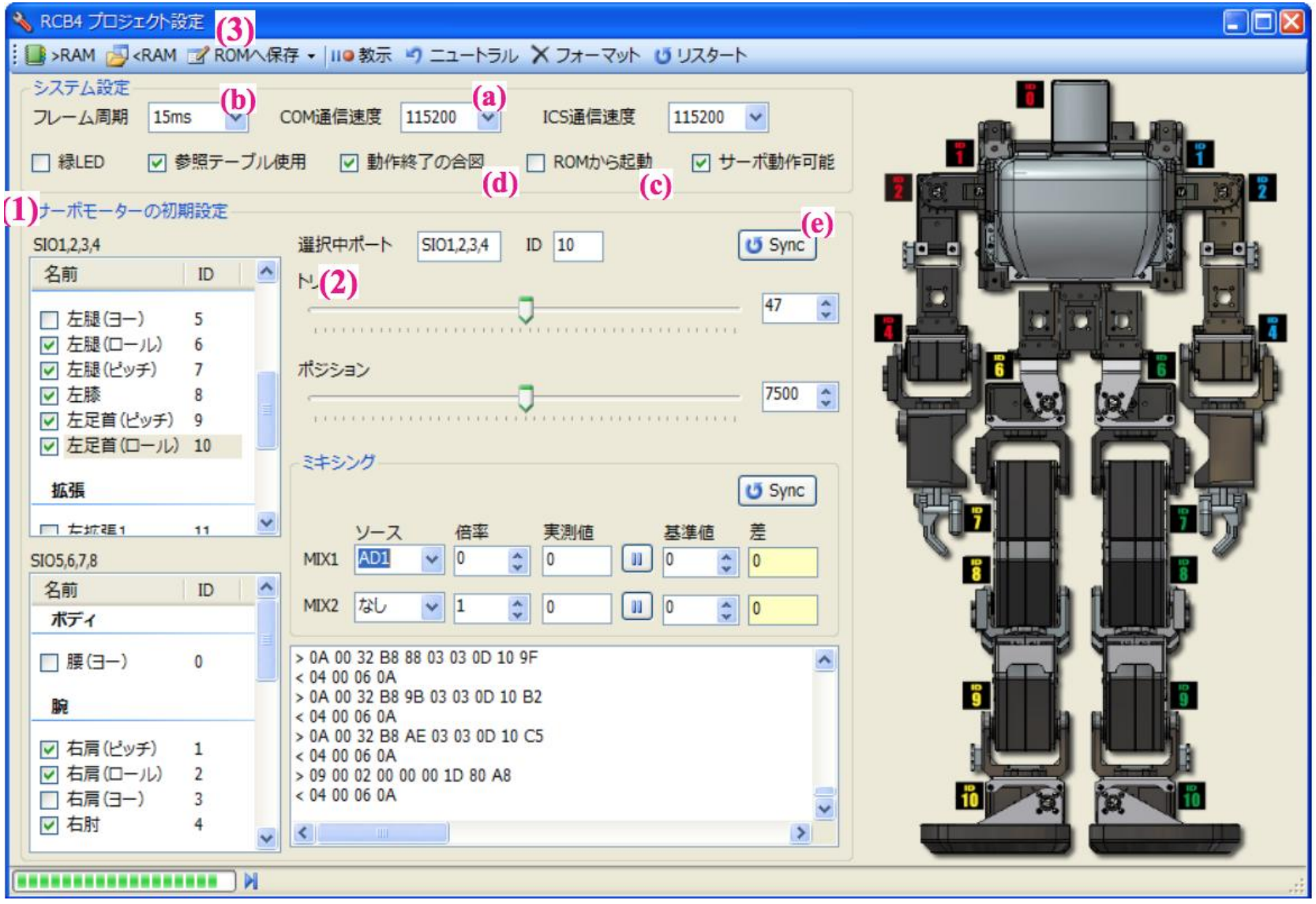


life.augmented

Main Issue: 日本語が分かりますか？

- Original Heart-to-Heart 4 proprietary Software was in japanese. Impossible to use.
- Also it was based on a Windows PC version, who required a wired connection to the robot





life.augmented

First Objectives… and difficulties

- Bluetooth Connection:
  - Create the wireless communication system to avoid wired (and dangerous) connection

But……

- How to handle the software without understanding japanese?
- How to understand which type of signal is transmitted on the wired connection?
- Need of a communication board lighter enough to not compromise the barycenter of the robot
- How to use ICS 3.5 protocol?

- Android App
  - Create an Android App to pilot the robot

But…..

- How to understand which commands must be passed to the robot?
- If we are able to understand the commands, how will be the best interface to command the robot from the touchscreen?
- How to establish a communication with the robot?

Commands Syntax

- The sequence of movements created by HTH4 is saved in a xml motion file.
- In this xml file all the position the servos have to assume are expressed through hexadecimal arrays.

```
<ProgramCode>
  <anyType xsi:type="xsd:string">2B 10 3D F3 3F 00 00 2D 4C 1D 7C 1A 97 10 2C 1E 65 18 56 14 1C 25 4C 1D 4C 1D B0 1D E8 1C 4C 1D 4C 1D B0 1D E8 1C 4C 1D 4C 1D 07</anyType>
</ProgramCode>
<Hint>ポジション設定</Hint>
```

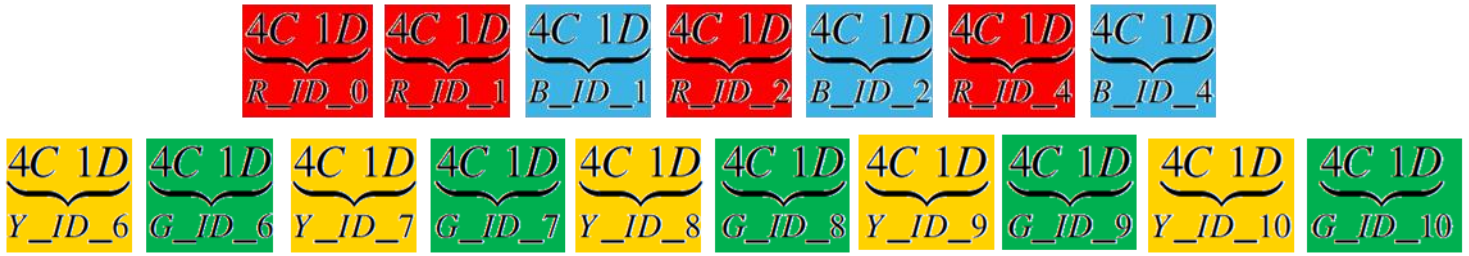
- To set the position of each servo, a hexadecimal array as the following is used:

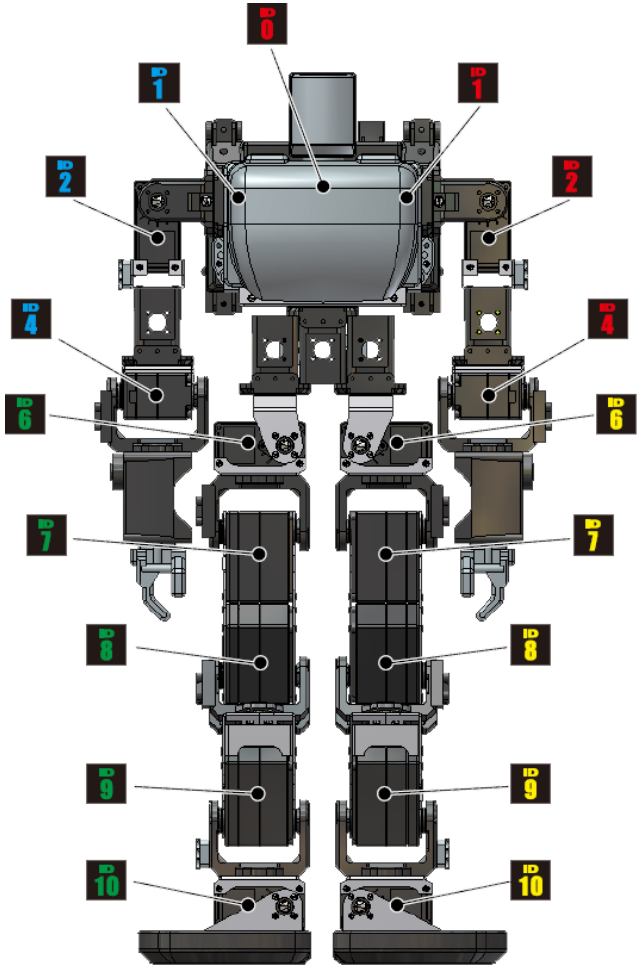
  2B 10 3D F3 3F 00 00 64 4C 1D 4C 1D 4C 1D 4C 1D 4C 1D 4C 1D 4C 1D 4C 1D 4C 1D 4C 1D 4C 1D 4C 1D 4C 1D 4C 1D 4C 1D 4C 1D 4C 1D 07

  - 2B is the total number of bytes in the array.
  - 10 3D F3 3F 00 00 is a preamble which function is not clear.
  - 64 is the frame number.
  - the pairs 4C 1D indicate the position which is obtained inverting the positions of the bytes. For example, in this case the hexadecimal value to consider is 0x1D4C which corresponds to the decimal value 7500.
  - 07 is a checksum. It is the remainder of the division between the sum of all the hexadecimal values and the value 0x100 (256 in decimal). This value is used to check the correctness of the array

ID configuration

- With reference to figure, after assigning an unique ID to each servomotor, there are the following correspondences between the position hexadecimal pairs and the IDs



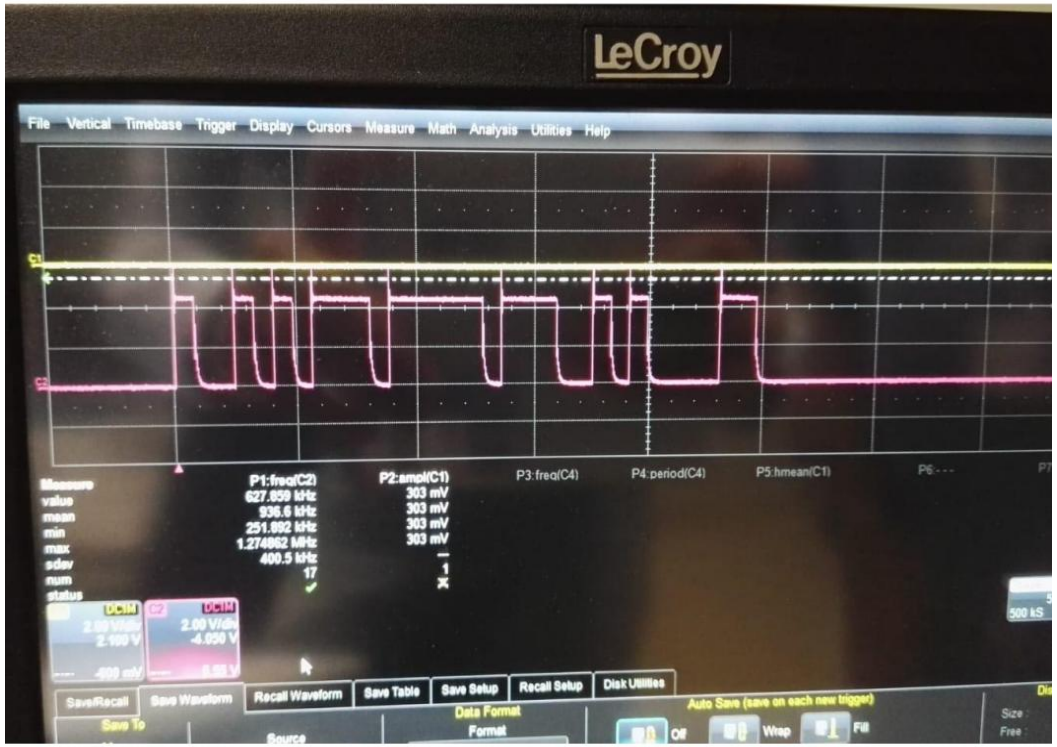


- Assuming that the angle of rotation of the servo motors is 270°

Angle	Position	HEX
-135°	3500	AC 0D
0°	7500	4C 1D
+135°	11500	EC 2C

life.augmented

Signal Snifing

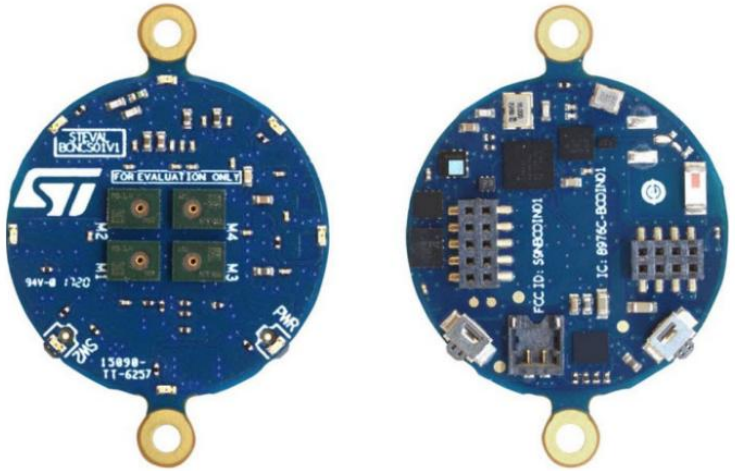


- It was therefore considered appropriate to detect the signal sent to the robot from the serial port.
- The same hexadecimal string used in H2H4 was sent using a SW called “Advanced Serial Port Monitor”.
- The signal was captured with the oscilloscope.
- An analysis showed that the signal was similar to USART, but mirrored with respect to the horizontal axis. In addition, the logic levels were between 0 and 5 Volts.

STEVAL-BCNKT01V1 (BlueCoin)

- The BlueCoin board lets explore advanced sensor fusion and signal processing functions for robotics and automation applications with a 4 digital MEMS microphone array, a high-performance 9-axis inertial and environmental sensor unit and time-of-flight ranging sensors.
- A high-performance STM32F446 180 MHz MCU enables real-time implementation of the very advanced sensor fusion algorithms like adaptive beamforming and sound source localization, with ready-to-use, royalty-free building blocks.
- The BlueCoin can **connect via the on-board BLE link to any IoT** and smart industry wireless sensor network.
- To be able to use this board we need to invert the signal on the UART port and to increase the logic level from 0-3.3V to 0-5V.

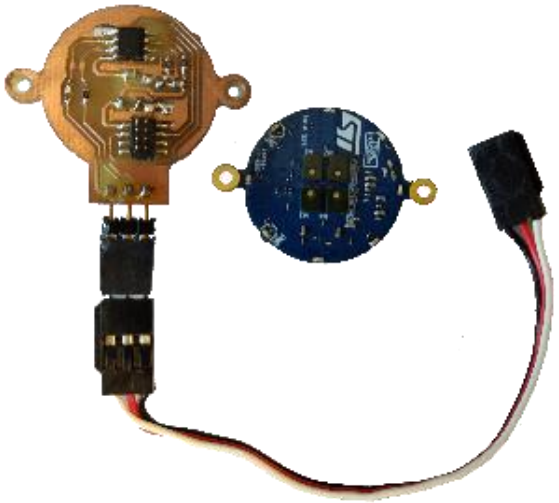
Low Energy

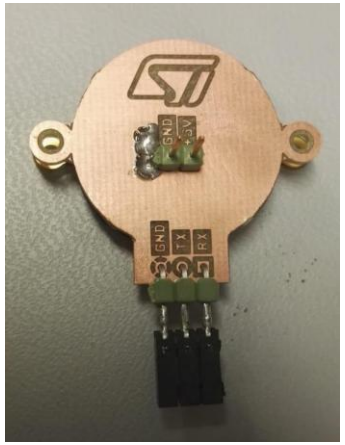


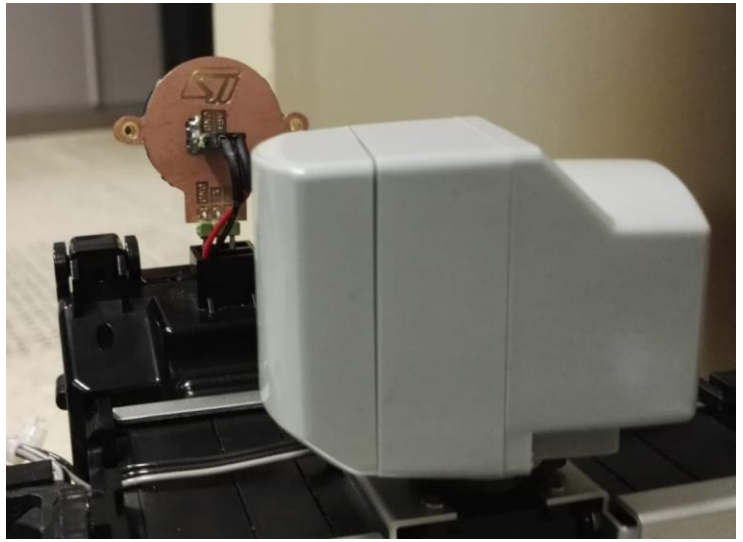
life.augmented

Inverter 485

- A board with an inverter 485 has been designed and built, powered directly through the 5V power supply of the board placed in the robot's backpack, and which adapts perfectly to the size of the BlueCoin.
- Using the Bluetooth embedded into the BlueCoin, the wireless communication with the robot is enabled







FW BlueCoin

- It was therefore necessary to manage both the interaction via Bluetooth between the app and the control board, and the interaction via UART between the board and the robot.
- The original ALLMEMS1 package delivered with BlueCoin has been modified to insert the `DebugConsoleCommandParsing()` function into the file `sensor_service.c`.
- A protocol has been defined to fix the syntax of commands sent/received to/from the Android app.

```
static uint32_t DebugConsoleCommandParsing(uint8_t * att_data, uint8_t data_length)
{
  uint32_t SendBackData = 1;

    /** Code added by Alessandro Lauria **/
    /** Check if the first word is "Bot", after check the command searched**/
    if(!strncmp("Bot:",(char *)(att_data),4)){
      bluetooth_command_arrived = 1; //avvisa il main che è arrivato un comando bluetooth
      char* msg = (char*)(att_data);
      char* command = "null"; //variabile dove si salva il messagio arrivato dopo la stringa "Bot:"
      command = strtok(msg,":");
      command = strtok(NULL," ");
      /*Confronto command con messaggi predefiniti ed eseguo quello richiesto*/
      /*Passa la stringa del comando e la mette in robot_command, che a sua volta leggerà il main */
      if(!strncmp("stop",command,4)){
        robot_command=command;
        robot_stop=1;
      }
      /*comandi joystick*/
      else if(!strncmp("upPress",command,7)){
        robot_command = command;
      }
      else if(!strncmp("downPress",command,9)){
        robot_command = command;
      }
      else if(!strncmp("leftPress",command,9)){
        robot_command = command;
      }
```

life.augmented

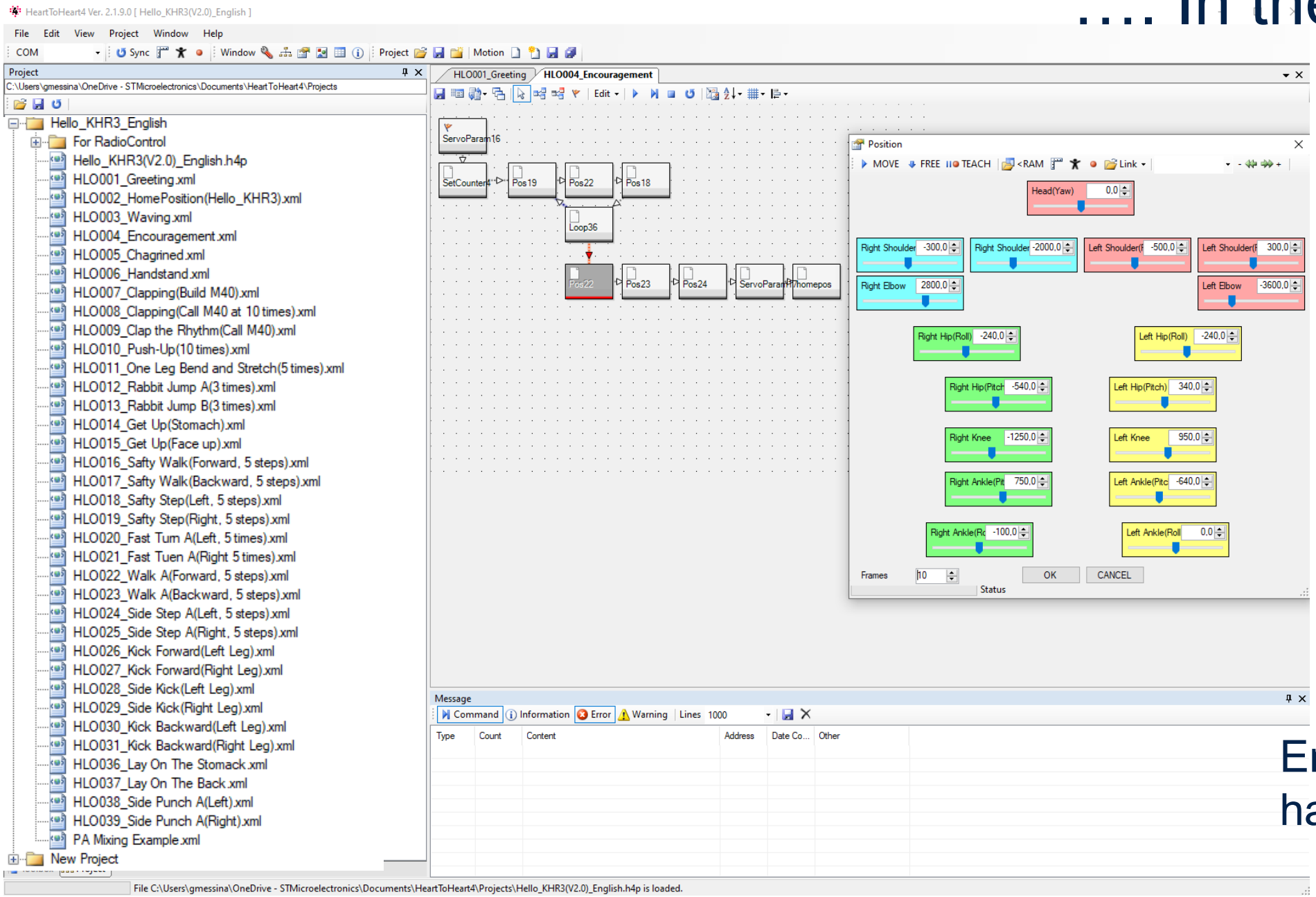
FW BlueCoin

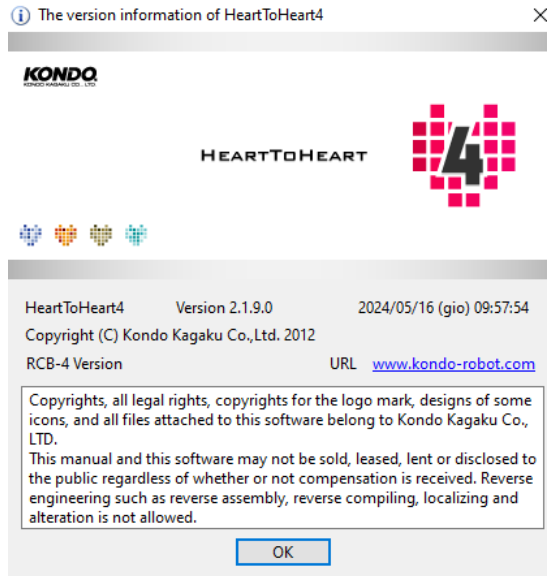
- The `Drive_Robot()` function compares the string passed as a parameter with preset strings corresponding to the default commands

```
else if(!strncmp("Greeting",command,8)){
  int delay = 1000;
  HomePosition();
  uint8_t data1[] = {0x2B,0x10,0x3D,0xF3,0x3F,0x00,0x00,0x50,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C
  ExecMovement(data1,sizeof(data1),1,delay);
  uint8_t data2[] = {0x2B,0x10,0x3D,0xF3,0x3F,0x00,0x00,0x14,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C
  ExecMovement(data2,sizeof(data2),2,delay);
  uint8_t data3[] = {0x2B,0x10,0x3D,0xF3,0x3F,0x00,0x00,0x64,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C
  ExecMovement(data3,sizeof(data3),3,delay);
  HomePosition();
  Command_Ended_Message();
}
else if(!strncmp("Layontheback",command,12)){
  int delay = 1000;
  HomePosition();
  uint8_t data1[] ={0x2B,0x10,0x3D,0xF3,0x3F,0x00,0x00,0x32,0x4C,0x1D,0x2C,0x1A,0x6C,0x20,0x34,0x21,0x64,0x19,0xA0,0x0F,0xF8,0x2A,0x4C,
  ExecMovement(data1,sizeof(data1),1,delay);
  uint8_t data2[] = {0x2B,0x10,0x3D,0xF3,0x3F,0x00,0x00,0x32,0x4C,0x1D,0x84,0x1C,0x14,0x1E,0x14,0x1E,0x84,0x1C,0x58,0x1B,0x40,0x1F,0x4C
  ExecMovement(data2,sizeof(data2),2,delay);
  uint8_t data3[] ={0x2B,0x10,0x3D,0xF3,0x3F,0x00,0x00,0x46,0x4C,0x1D,0x88,0x13,0x10,0x27,0x14,0x1E,0x84,0x1C,0x88,0x13,0x10,0x27,0x4C,
  ExecMovement(data3,sizeof(data3),3,delay);
  uint8_t data4[] = {0x2B,0x10,0x3D,0xF3,0x3F,0x00,0x00,0x3C,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C,0x1D,0x4C
  ExecMovement(data4,sizeof(data4),4,delay);
  HomePosition();
  Command_Ended_Message();
}
```

life.augmented

.... In the meanwhile

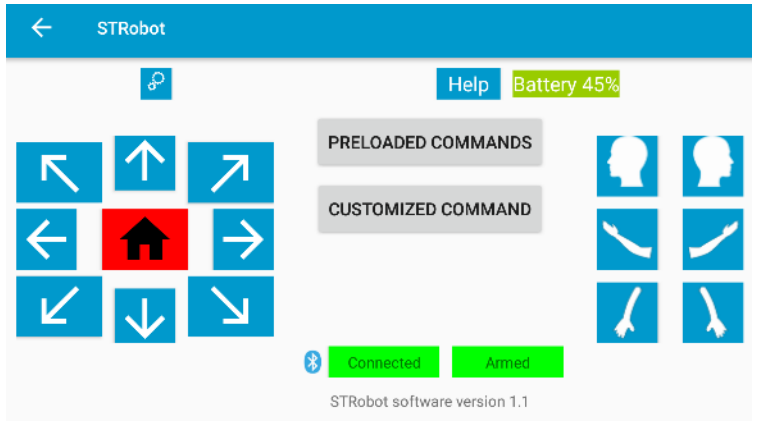




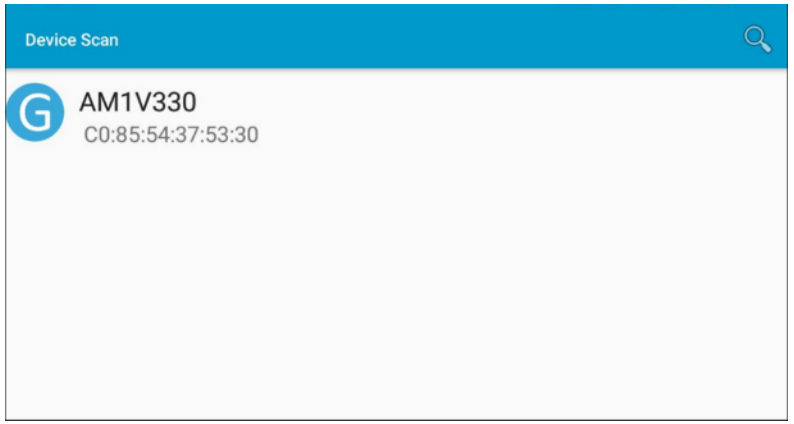
English Beta Version has been released!!!

App Android

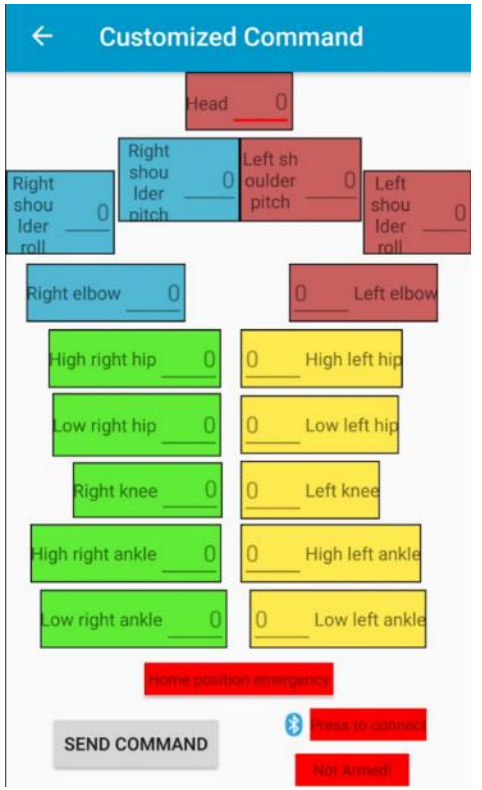
Main Screen



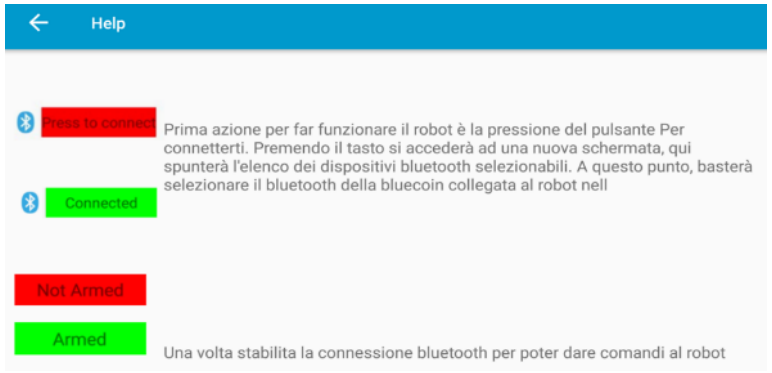
Bluetooth Scan



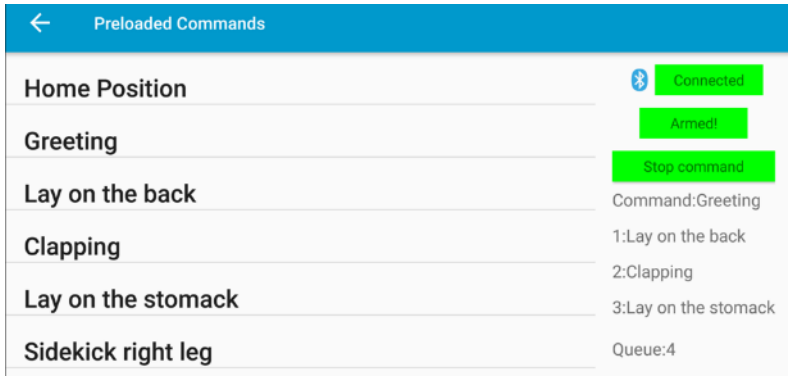
Custom Commands



Help

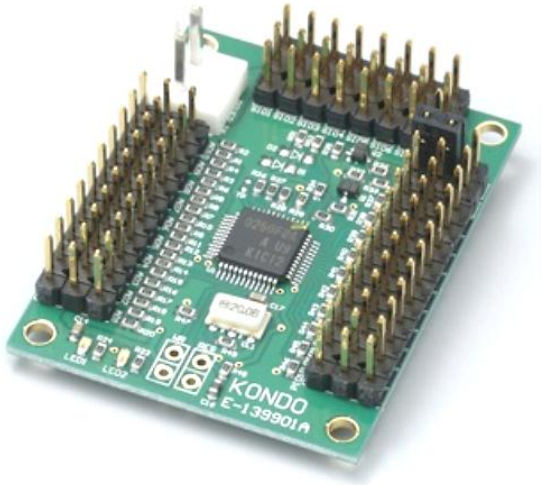


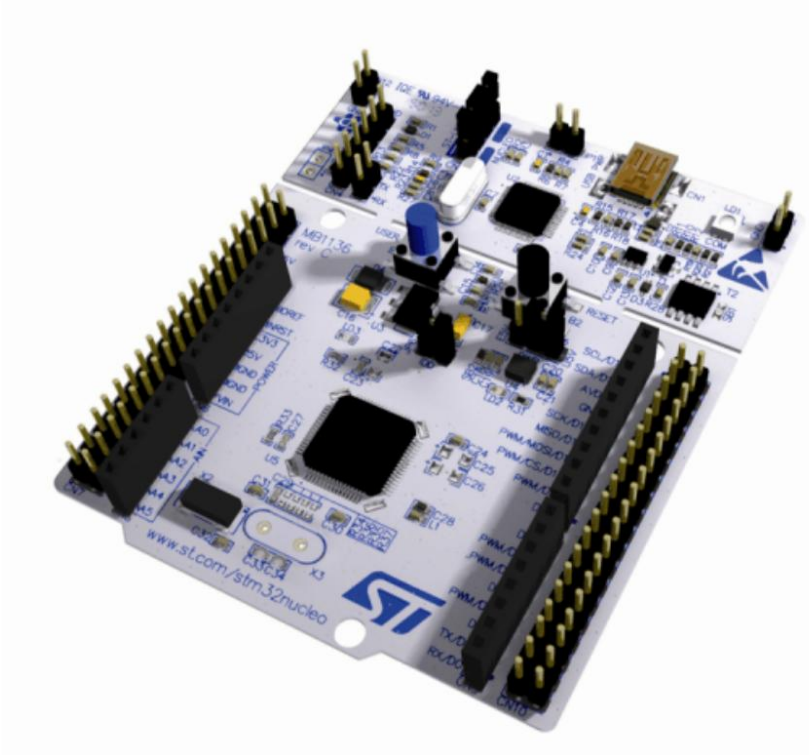
Preloaded Commands



life.augmented

One step Forward……

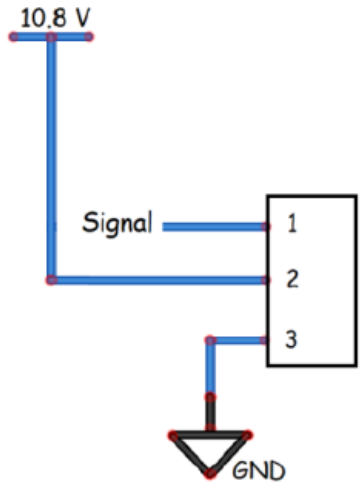




We need to move from Renesas board to ST Nucleo board!

life.augmented

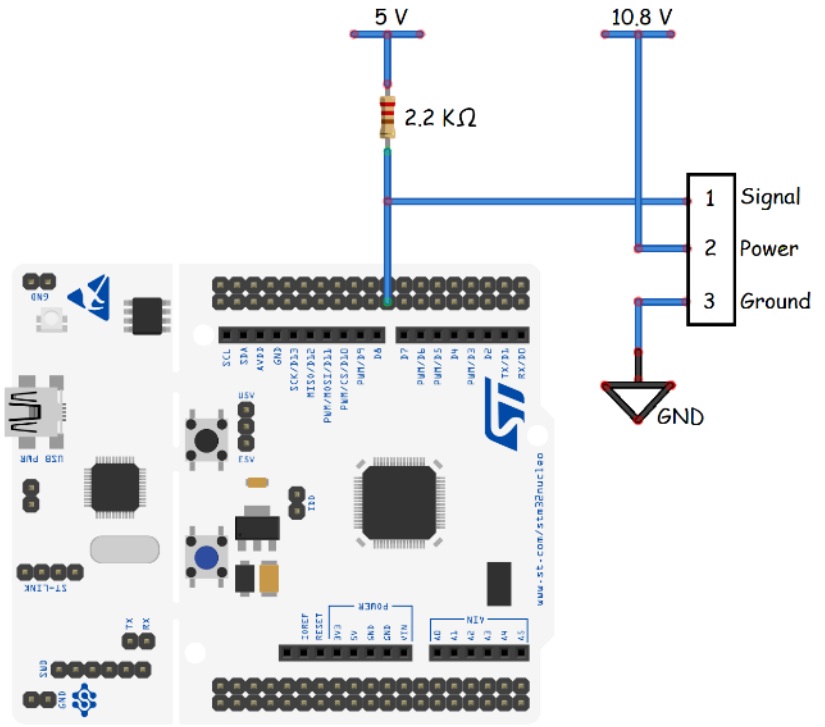
The Kondo KRS-2552RHV



- Communication standard: ICS3.5
- Three-wire connector :
  - the signal is applied to the number 1,
  - Vcc is connected to the number 2
  - Ground is connected to the number 3.
- Controlled using either a PWM or a serial signal
- Daisy Chained serial protocol allows the connection of several servos in single chain reducing the number of cables and simplifying connection.
- It is possible to assign an ID to each servo from 0 to 31.
- Daisy-chaining a large number of servos may introduce power supply issues.

ICS3.5 Communication Standard

- High-speed communication of up to 1.25 Mbps
- Max 32 servo motors connected on the same line
- Single data line used for serial communication (Half-Duplex)
- Multi-drop connection
- Signal level 5 V TTL
- ID Setting
- Read and Write Parameters
- Setting the Location

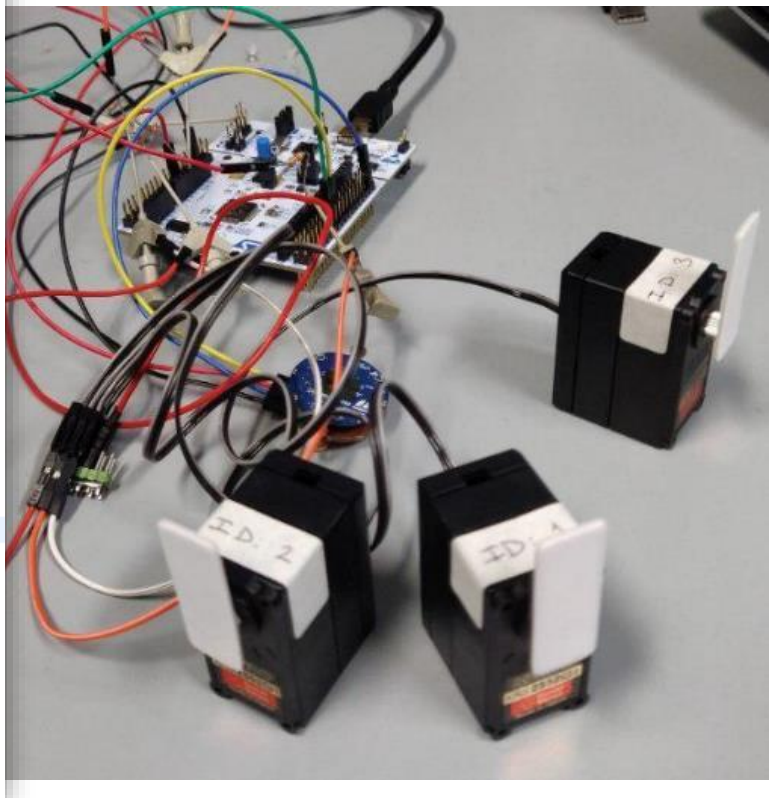


CMD	SC	DATA
Command Header + ID	Sub-control	Data

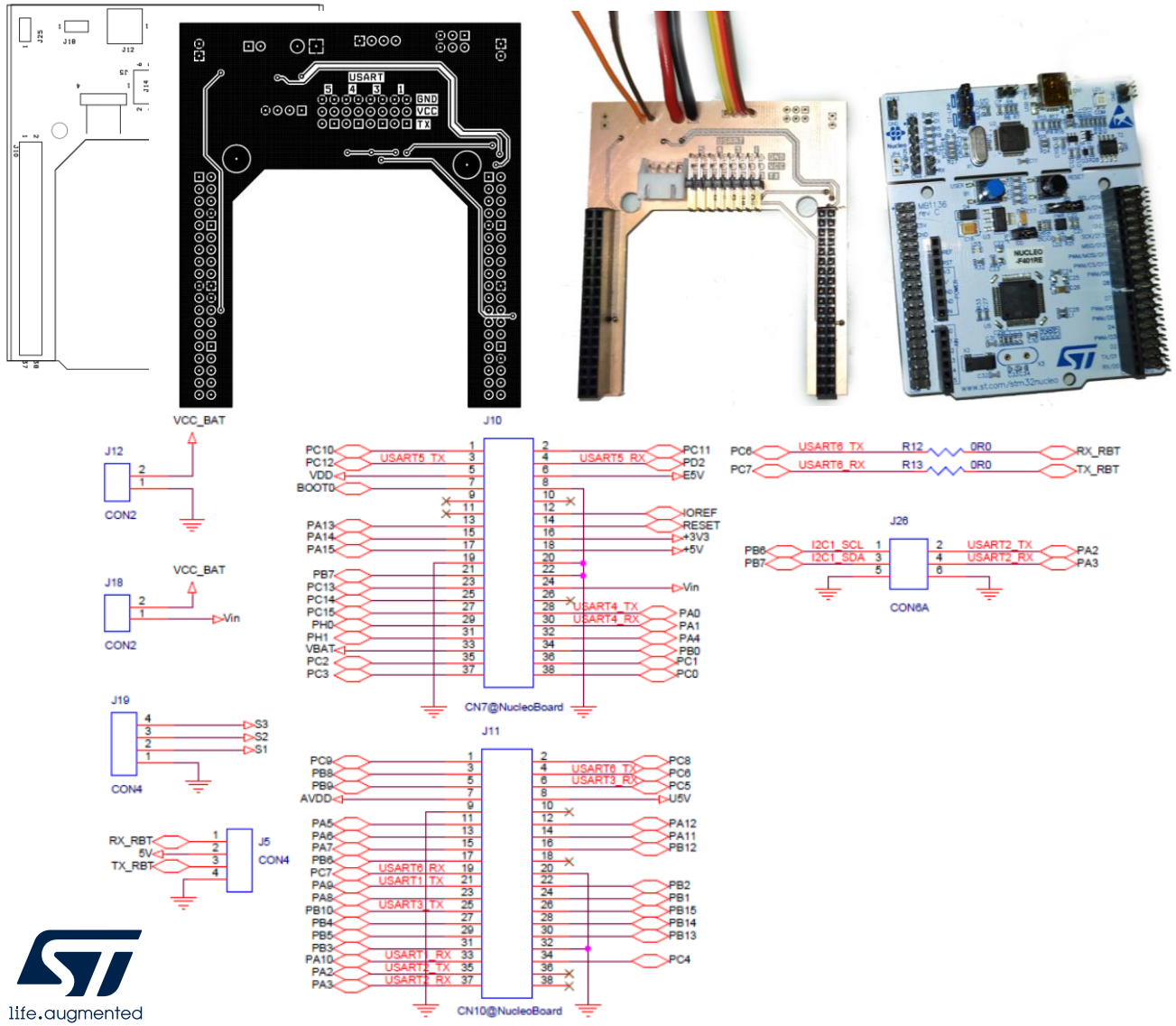
Firmware Implementation

- ICS.h:
  - ics_get_id()
  - ics_set_id()
  - ics_get_speed()
  - ics_set_speed()
  - ics_get_stretch()
  - ics_set_stretch()
  - ics_get_current()
  - ics_pos()
  - ics_free()

```
uint16_t ics_pos(ICSData *r, uint16_t id, uint16_t pos, UART_HandleTypeDef *huart)
{
    uint16_t byte_in = 3;
    uint16_t byte_out = 4;

    if(id>31)
    {
        snprintf(r->error,128,"ERROR: Invalid servo ID (0-31)");
        return -1;
    }

    if(pos>16383)
    {
        snprintf(r->error,128,"ERROR: Invalid servo position (0-16383)");
        return -1;
    }

    //build command
    r->swap[0] = id | ICS_CMD_POS;    //Command
    r->swap[1] = (pos >> 7) & 0x7F;    //POS_H: high 7 bits of pos
    r->swap[2] = pos & 0x7F;    //POS_L: low 7 bits of pos

    //send command
    HAL_UART_Transmit(huart, r->swap, byte_in, ICS_POS_TIMEOUT);

    for(int i=0; i < byte_in; i++)
        r->swap[i] = 0;

    //wait for the the response
    HAL_UART_Receive(huart, r->swap, byte_out, ICS_POS_TIMEOUT);

    //return the position
    return (((r->swap[2] & 0x7F) << 7) | (r->swap[3] & 0x7F));
}
```



Custom Nucleo Expansion Board

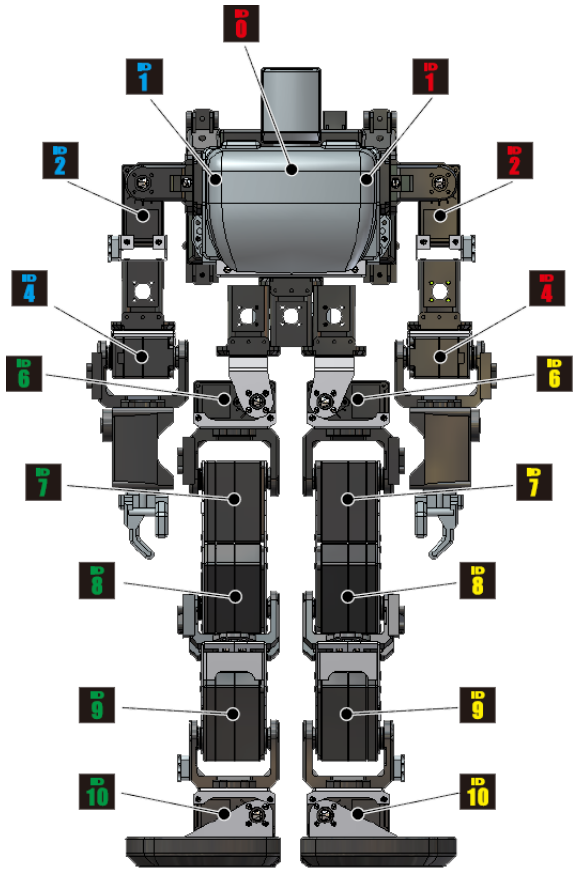


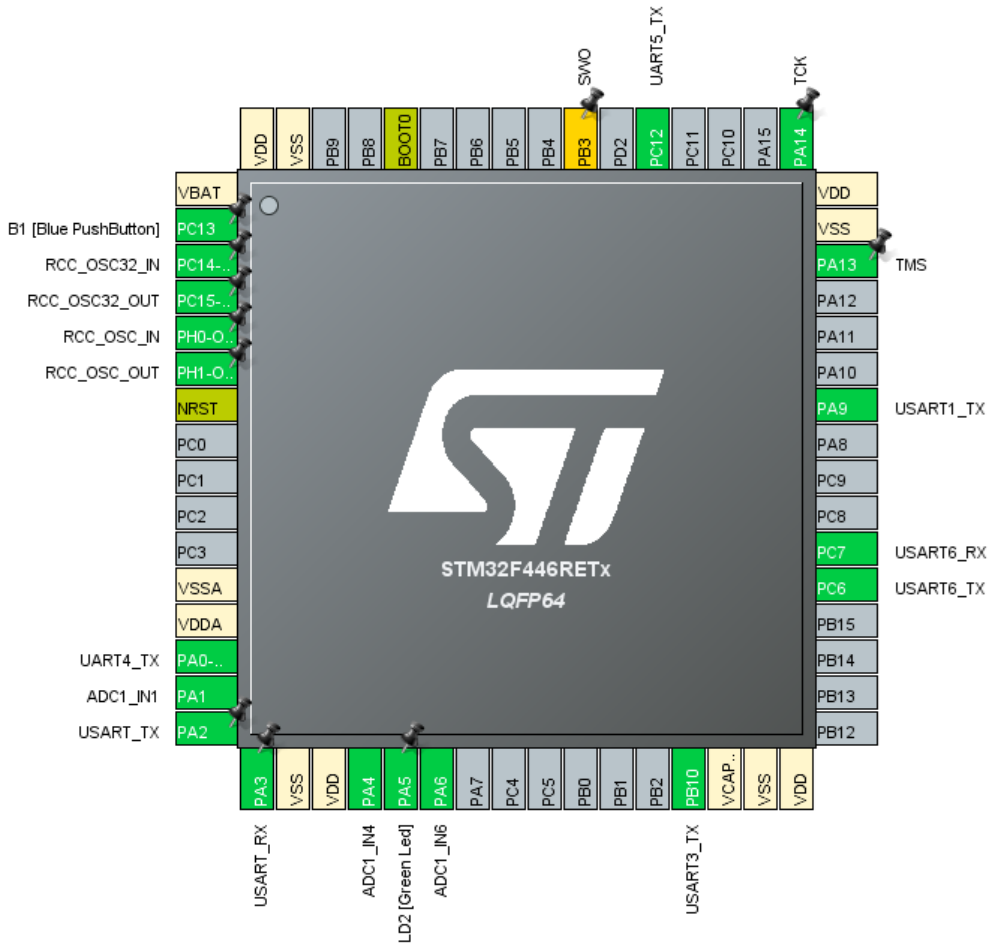
- The Nucleo requires a customized board to be mounted on the robot.
- This adapter includes all the resistors and the connectors :
  - serial communication between the Nucleo the servomotors
  - battery management
  - connection between the Nucleo and the Bluecoin.

life.augmented

STM32CubeMx

- Peripherals:
  - **USART1 → Red Line**
  - **USART3 → Blue Line**
  - **UART4 → Yellow Line**
  - **UART5 → Green Line**
  - USART2 → Debug USB
  - USART6 → Communication with Bluecoin
- ADC1 → Battery Management

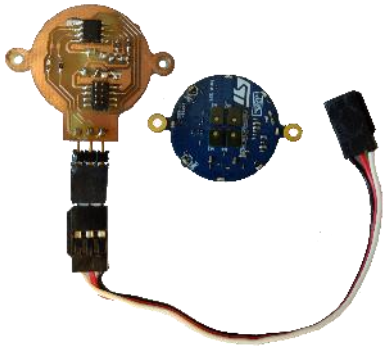




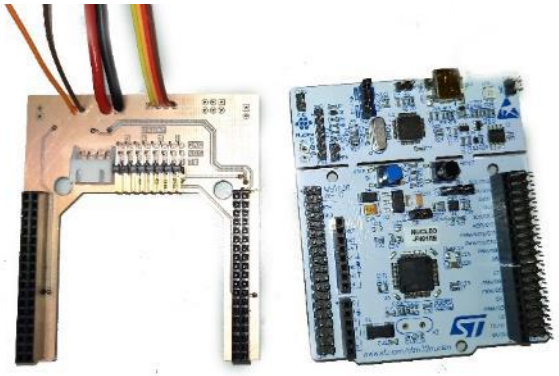
life.augmented

Finale Customizations

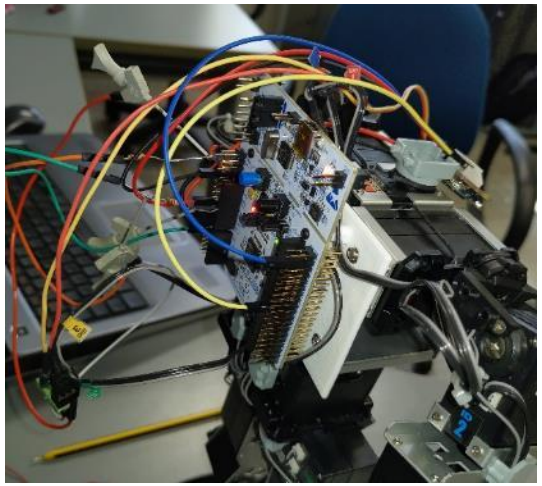
- Two expansion boards have been developed to connect the BlueCoin board and the Nucleo
  - For the BlueCoin, we used an expansion that allows to connect data and power from the Nucleo.
  - For the Nucleo board we have developed an expansion that allows the connection of 6 data channels, power supply and control of the battery.
- We could have done that without these expansions, but it would have been a lot less tidy.



Bluecoin and transceiver



Nucleo and Motors Connector

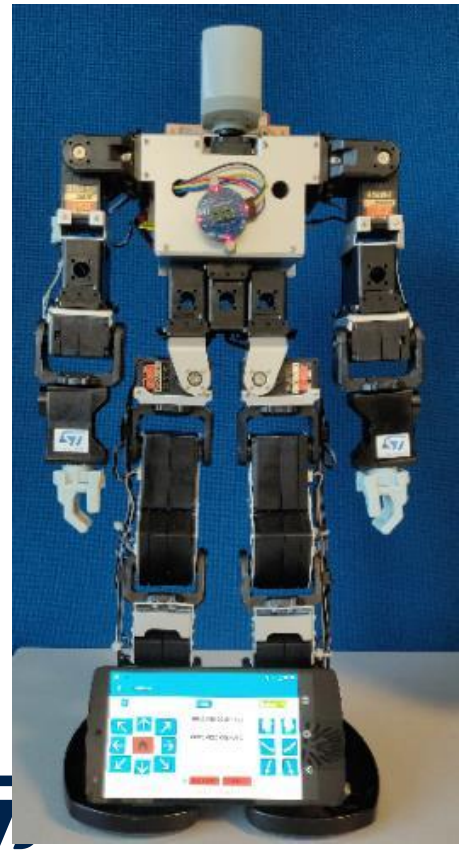


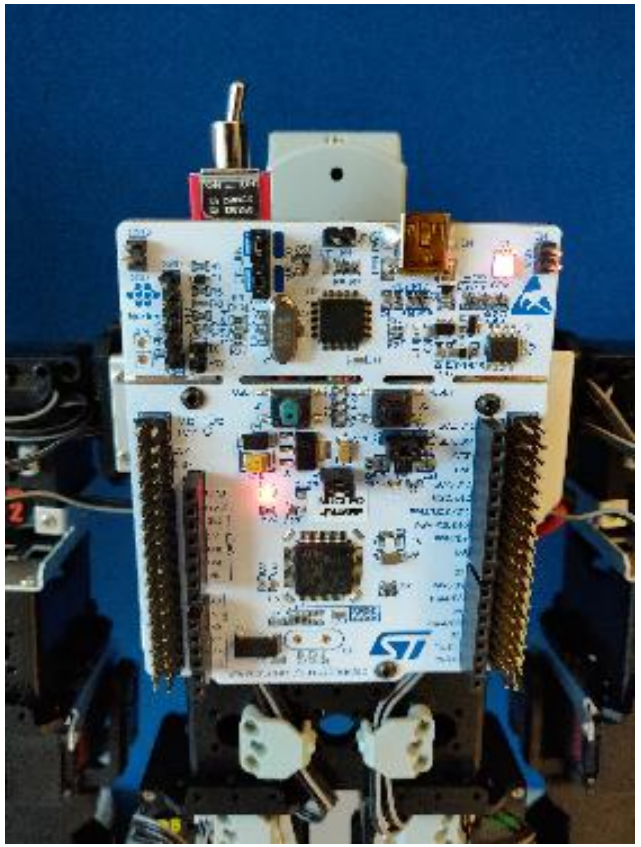
No custom expansion boards

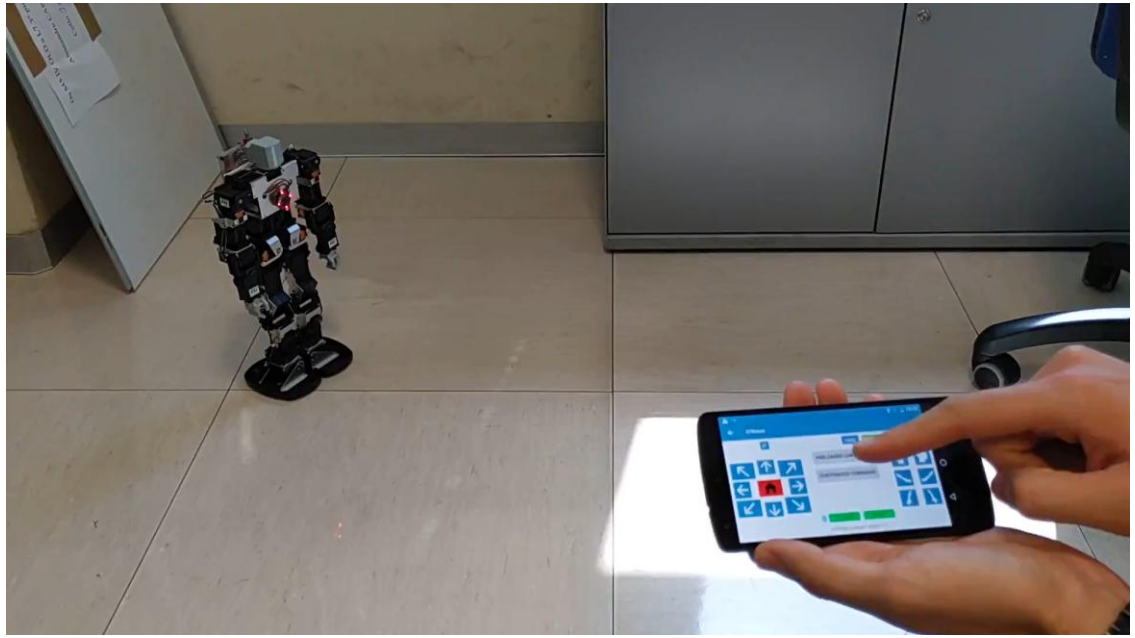
App Android

Kondo Robot with ST-Nucleo

- Replaced the Renesas-based main board for motor control with the Nucleo **STM32F44RE** board.
- Tested STM32-Kondo-Bot with paired **Android app.**







And Now?.....ISPU Project

- This experiment is part of a complete ISPU project in collaboration with University of Catania (prof. Santoro).
- The project will investigate three main research field:
  1. Improve the stability of a **humanoid robot**. The ISPU will be used to collect real-time data on the robot's movements. This data will be used to detect any imbalances in the robot and to take corrective actions.
  2. Estimate the drift of **Will-E robot** and adjust the trajectory. In this case, the ISPU sensor will be used to detect any loss of grip of the Will-E robot's wheels.
  3. Improve the stability and **flight control of a drone**. The ISPU sensor will also be used in drone applications to collect data on the drone's movements and to correct orientation.

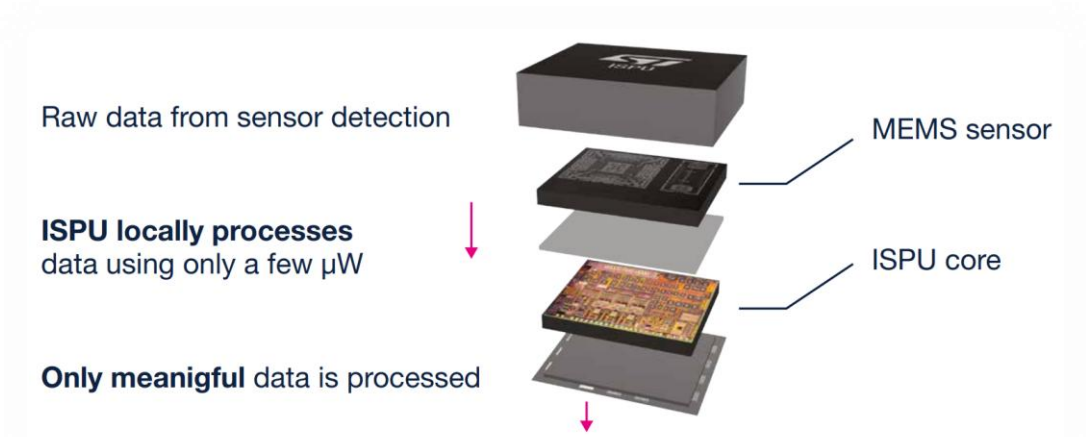
Intelligent Sensor Processing Units

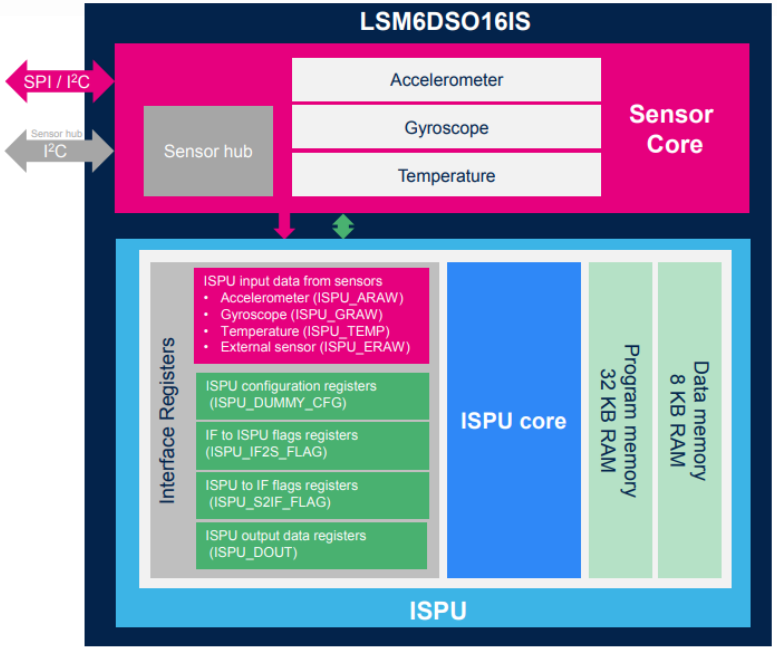
ISPUs introduce artificial intelligence into the world of sensors, within a compact and low-power device:

- The core reads the values generated by the sensors.
- It processes them using pre-processing algorithms or artificial intelligence.
- It provides the result via I2C or SPI interface.

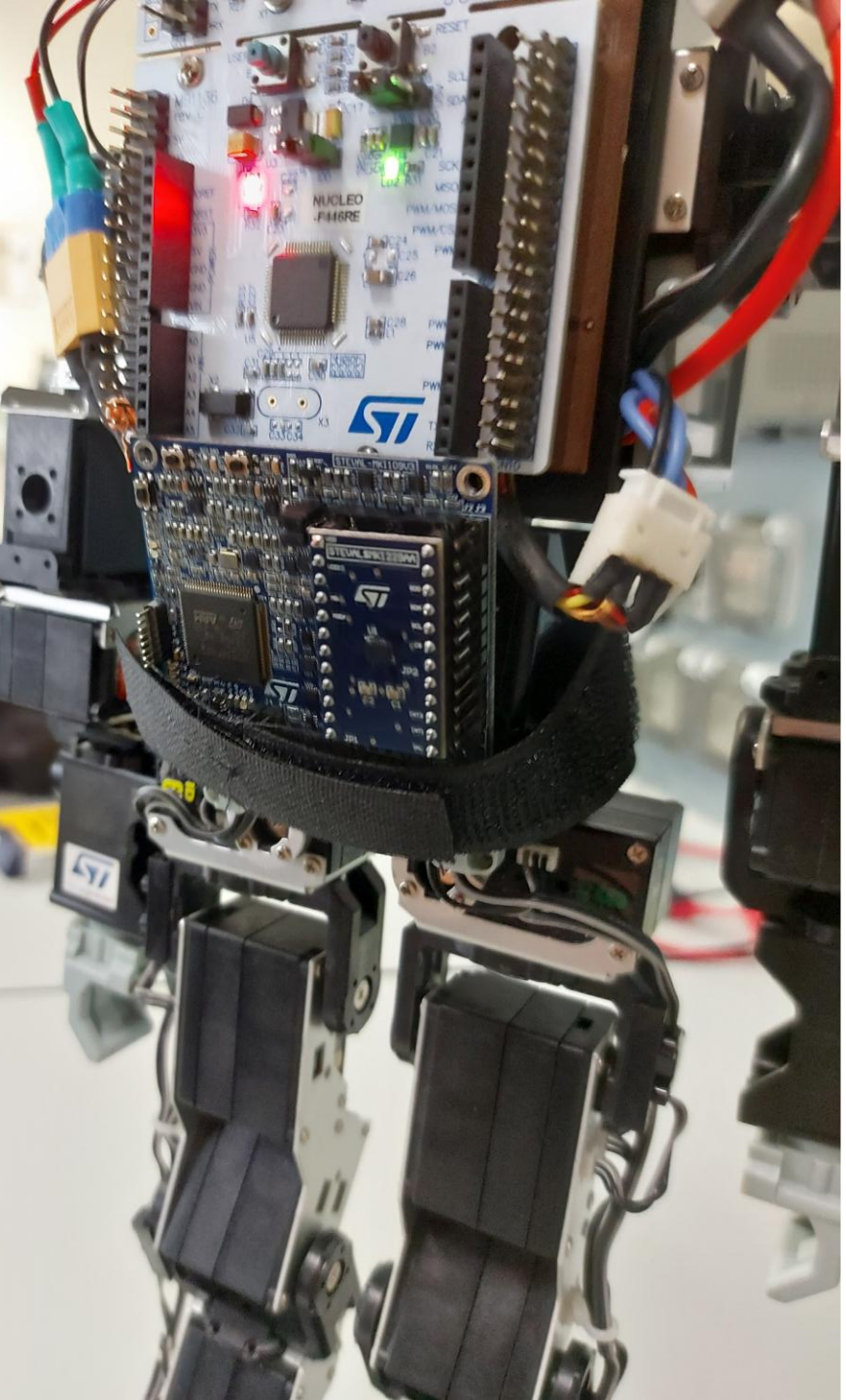
Used sensor: LSM6DSO16IS

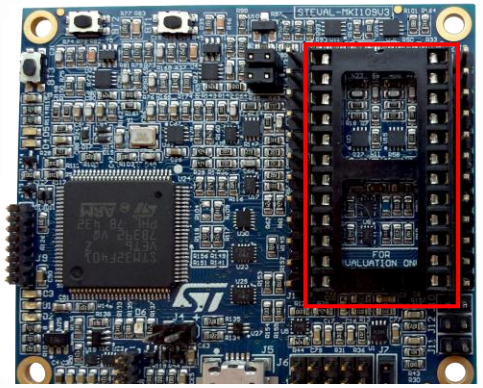
- Accelerometer and Gyroscope.
- **32KB of programmable RAM**.
- **8KB of data RAM**.





life.augmented

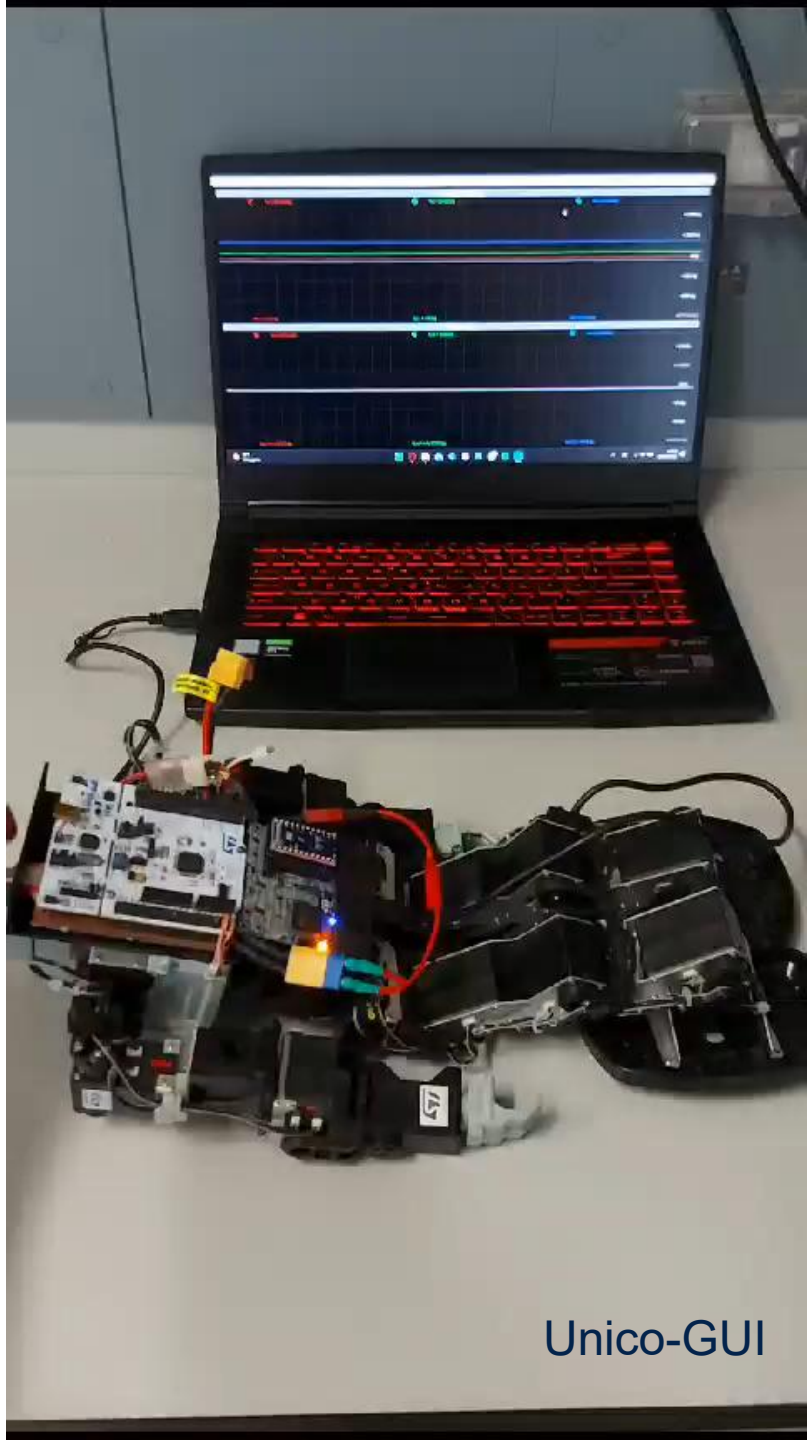


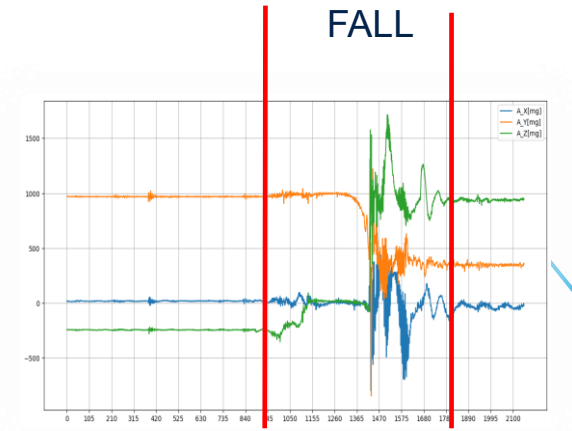


STEVAL-MKI109V3

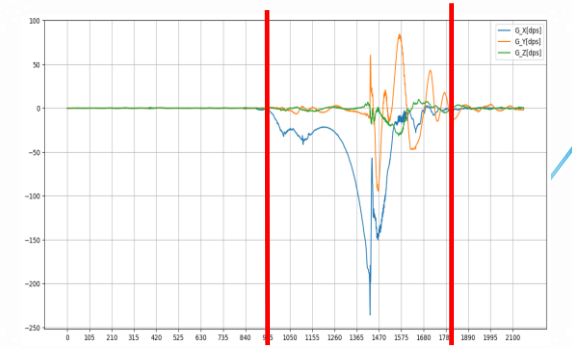
- DIL24 socket for communication with MEMS devices
- Equipped with a STM32F401
- Compatible with Unico-GUI
- **Difficult to customize**
- **Only accessible via Cable connection**
- **Difficult to acquire data**

Sensor Fusion

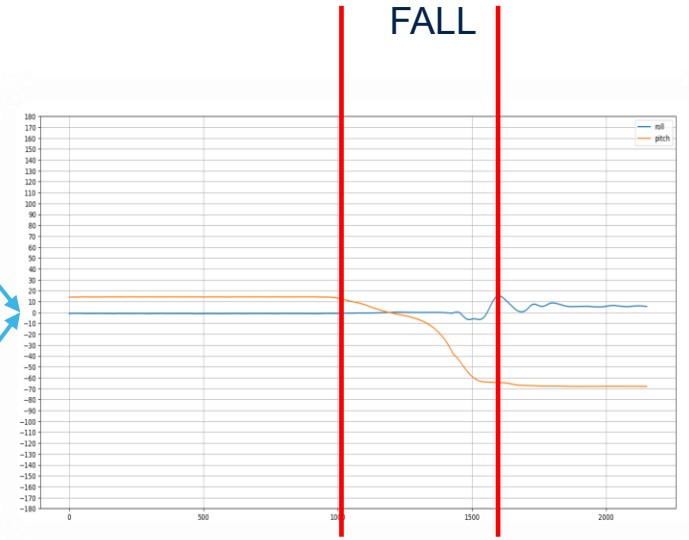




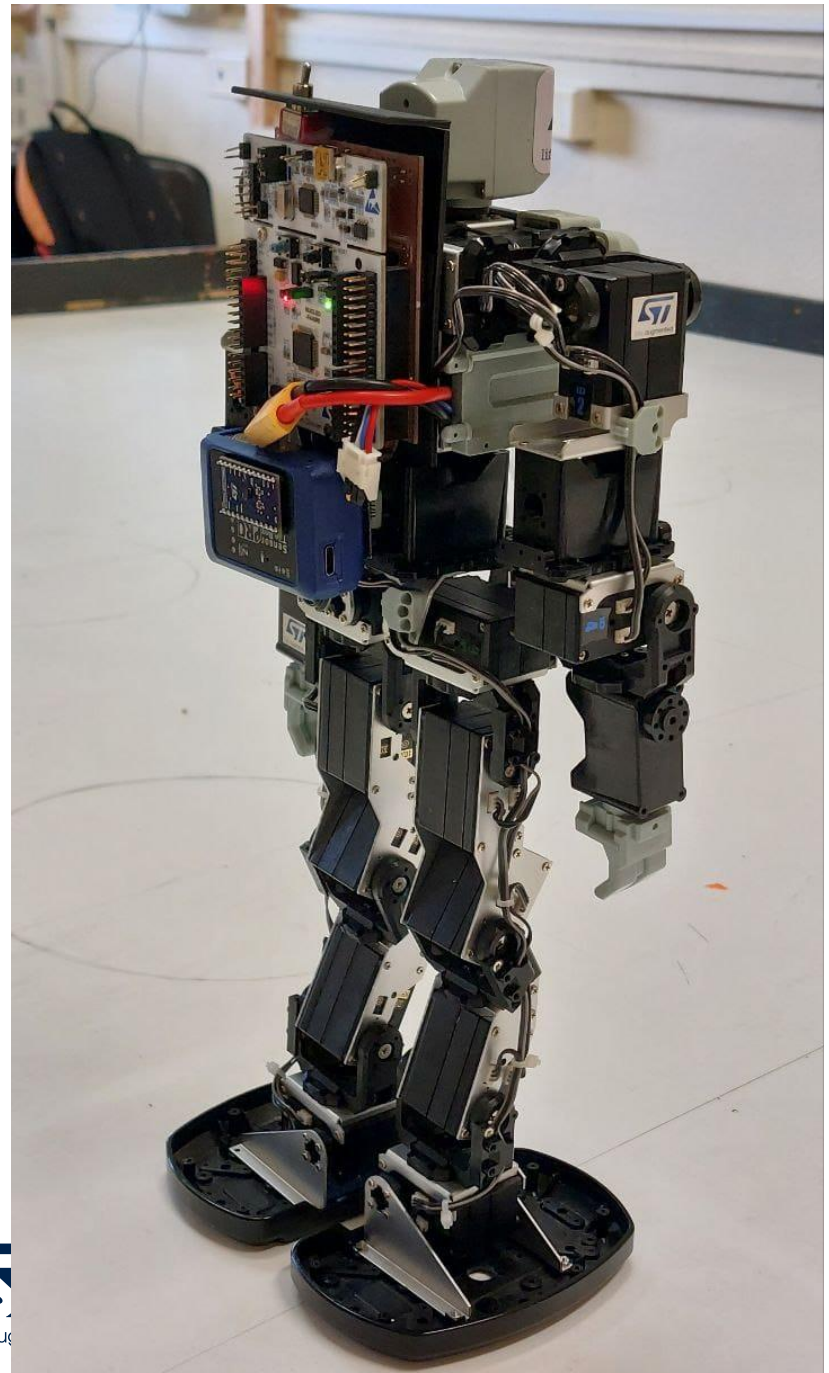
Data recorded by the accelerometer



Data recorded by the gyroscope



Roll and **Pitch** data obtained after sensor fusion filter application (**Madgwick filter**)



SensorTile.box PRO

- Socket DIL24
- **microSD card slot**
- Low-power STM32U585AI
- **BLE Module**
- High-precision sensors
- **Battery operated**

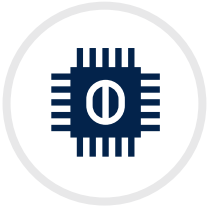
4 falling classes detection

life.augmented

Dataset construction

Step 1

Integration of the Madgwick filter within the ISPU



Step 2

Implementing Registration Firmware for the SensorTileBox.Pro

Step 3

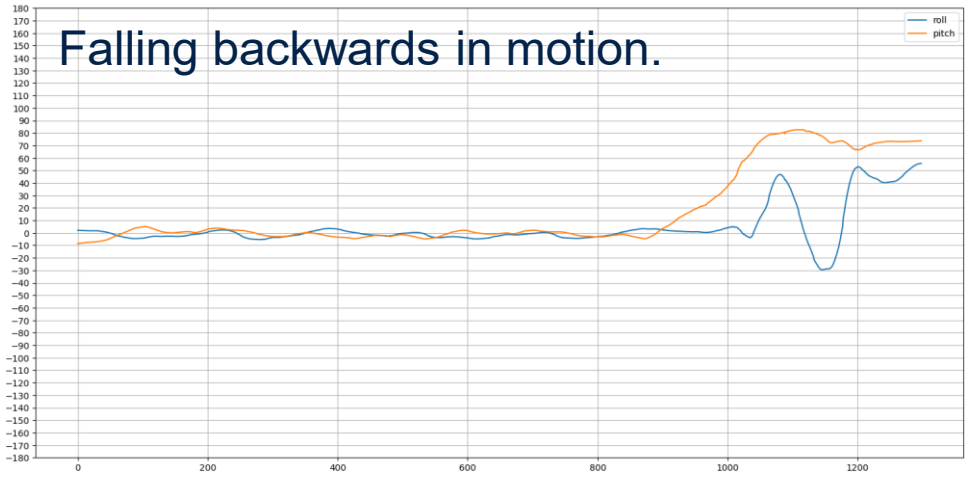
Capturing roll and pitch data

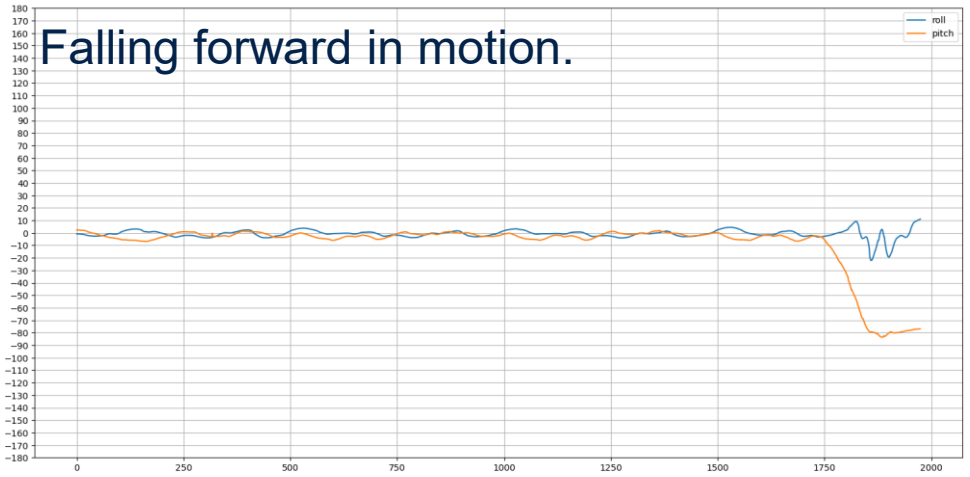
Step 4

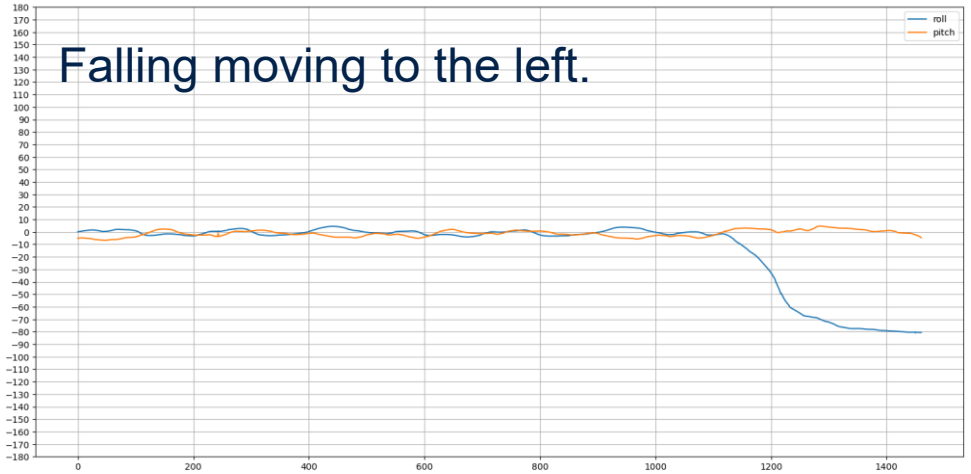
Dataset labeling

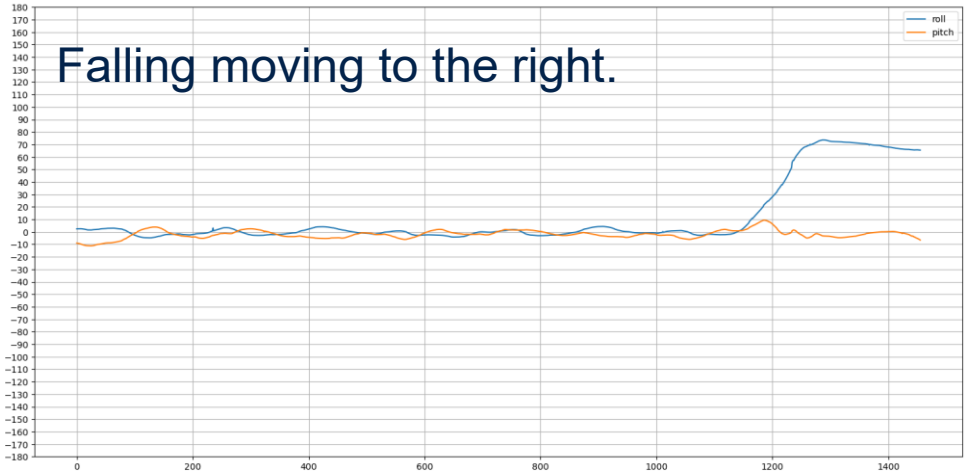
life.augmented

Falls with moving robots





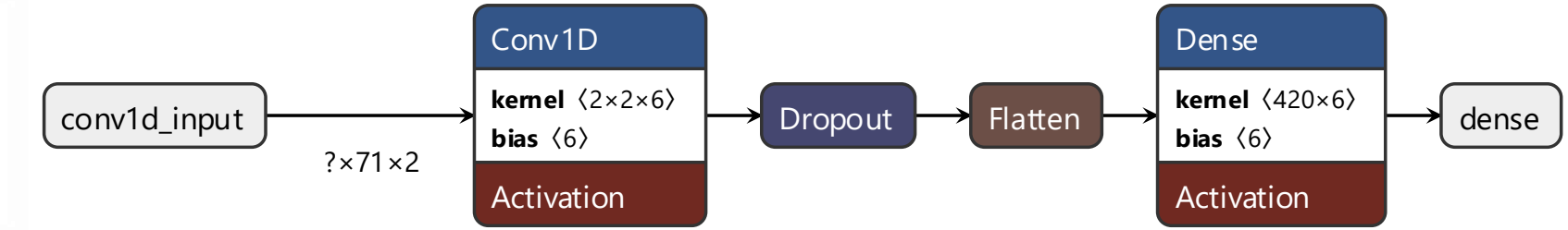




Dataset acquisition: 20 sequences for each classes (stationary, walking, front_falling, right_falling, back_falling, left_falling) of about 250 samples.

life.augmented

First model developed



- The model is inspired by a Human Activity Recognition network*
- The network has been significantly simplified to meet the ISPU's limited memory resource needs.
- The sampling rate used by the sensors is 416 Hz
- The network is structured as follows:
  - 6-filter convolutional layer with ReLU activation function and 71 × 2 input shape
  - Dropout layer with a factor of 0.5, to prevent overfitting
  - Flatten layer
  - Strongly connected output layer with Softmax activation function

*Human Activity Recognition: https://github.com/STMicroelectronics/stm32ai-wiki/blob/master/AI_resources/HAR/Human_Activity_Recognition.ipynb

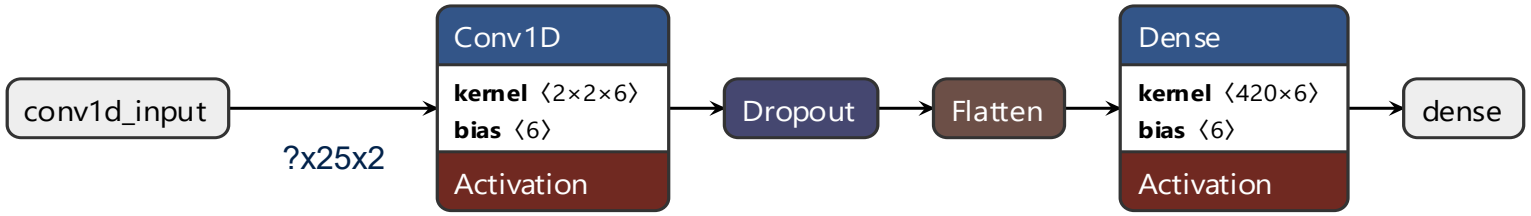
Results

Accuracy around 93%

75% Training set, 25% Test set

life.augmented

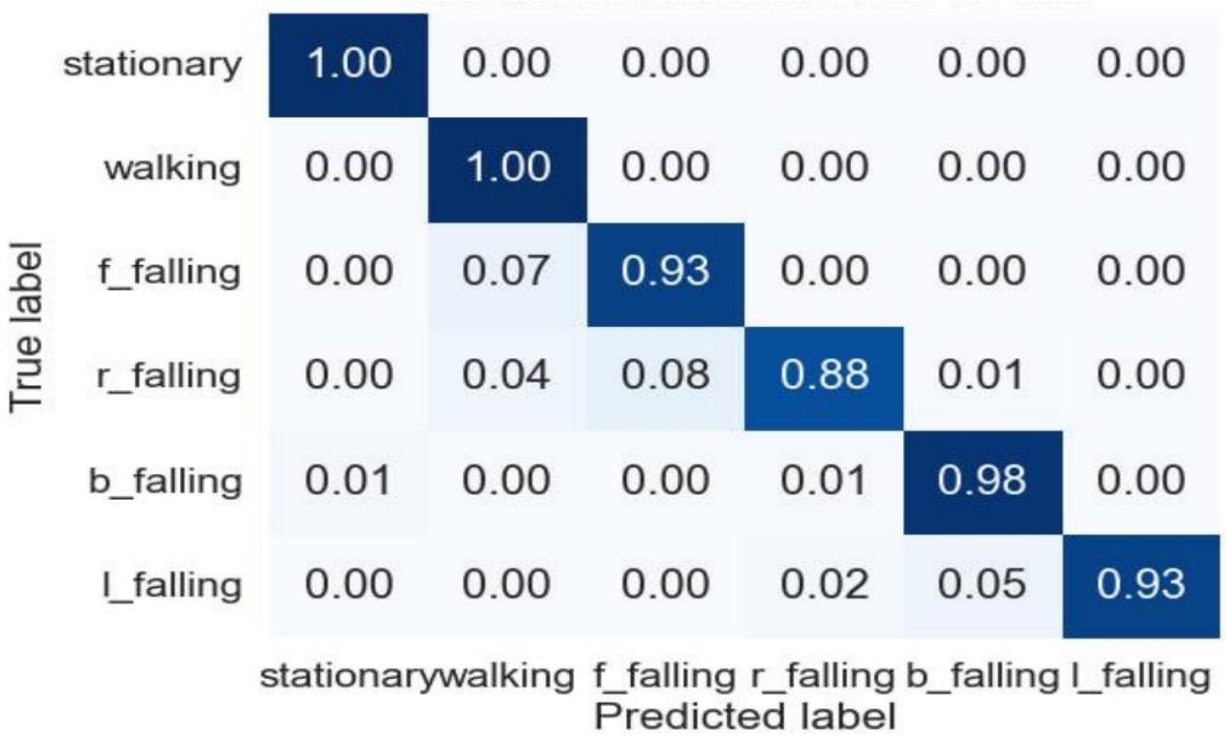
Second model developed



- Final release size of the **first model**, after compression and quantization + Magdwick filter (15KB) was **33000 bytes over 32768 bytes** available on ISPU.
- The network has been further simplified to meet the ISPU's memory and processing time:
  - The sampling rate used by the sensors has been reduced to **52 Hz (comprise network reaction time)**
  - The network is structured as follows:
    - 6-filter convolutional layer with ReLU activation function and **25 × 2** input shape
    - Dropout layer with a factor of 0.5, to prevent overfitting
    - Flatten layer
    - Strongly connected output layer with Softmax activation function
    - Acquisition Dataset errors were removed
- Final release size of second model, after compression and quantization: **27828 bytes with an Accuracy of about 94%**

Final results

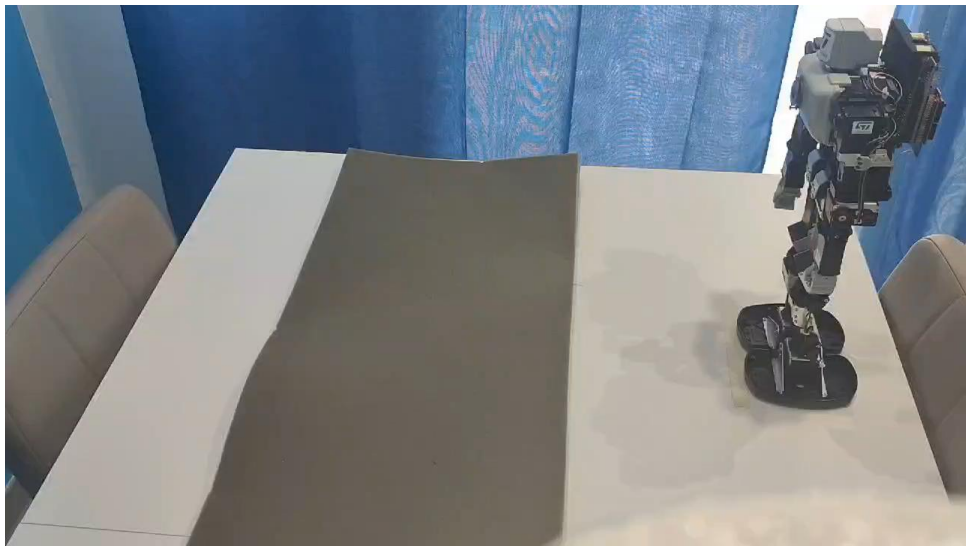
Accuracy around 94%

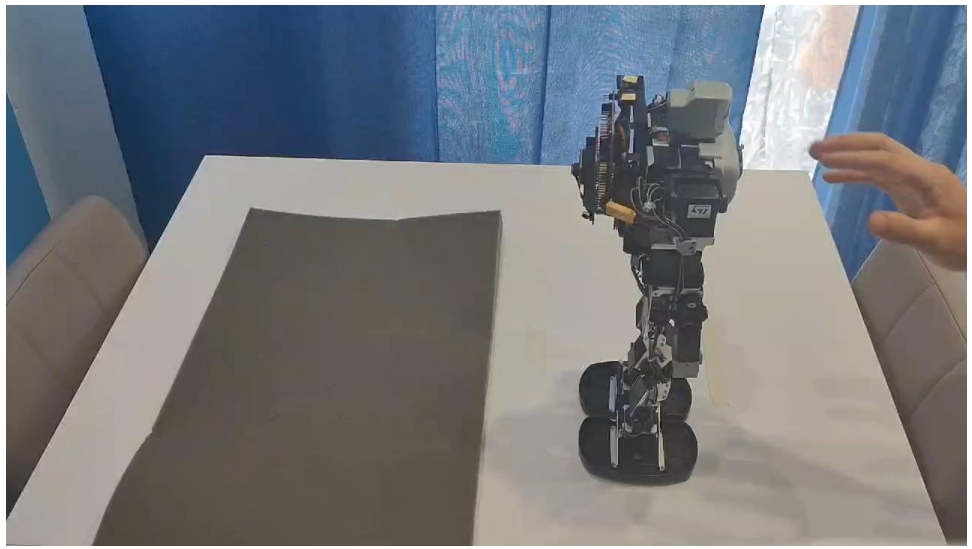


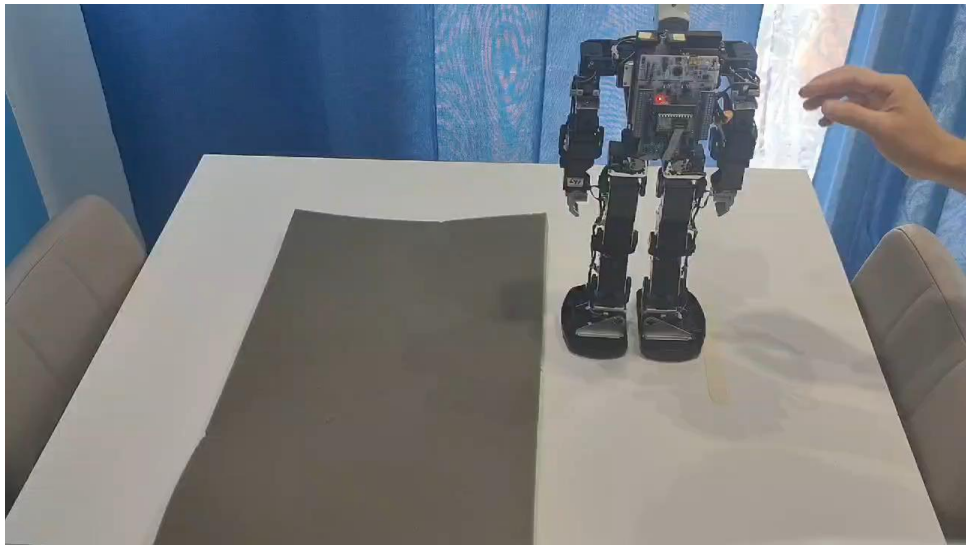
75% Training set, 25% Test set

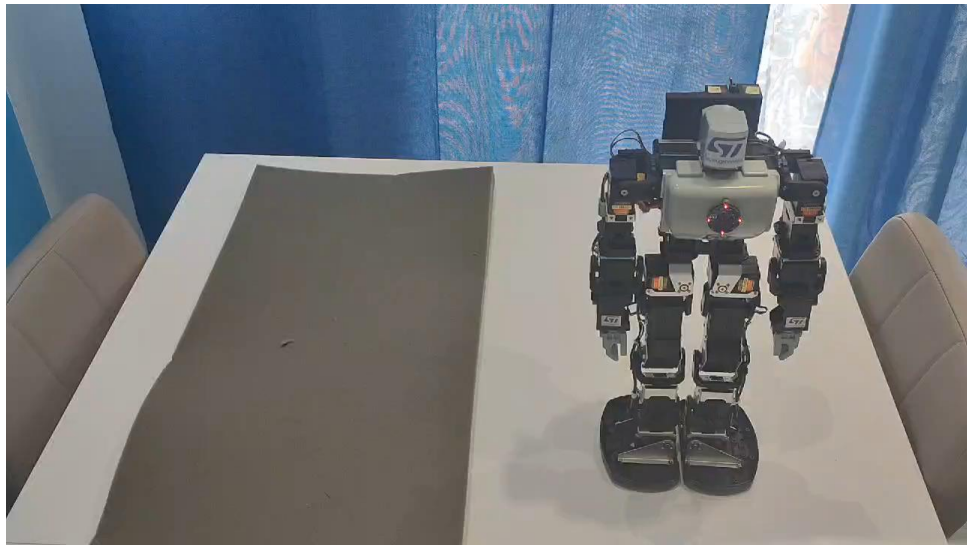
life.augmented

Falling + recover









life.augmented

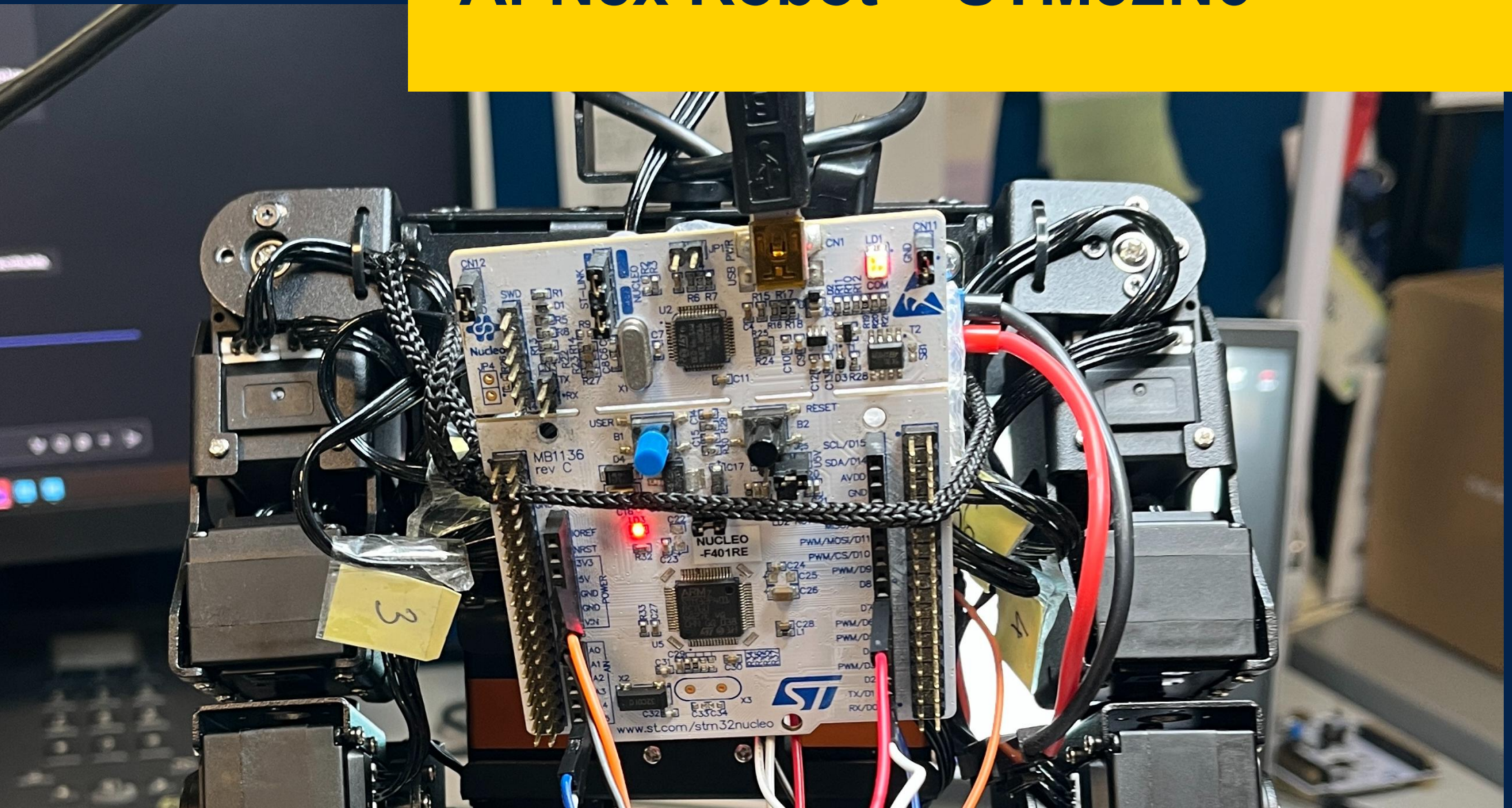
Conclusion

- Project Started in 2017, still ongoing!!
- UniCT referent: Prof. Corrado Santoro
- Students Involved:
  - Alessandro Lauria – Bluecoin FW (2017/2018)
  - Francesco Anastasio – Android App(2017/2018)
  - Mario Bonanno - Audio command Recognition (2019/2020)
  - ***Marco Beninato** - Nucleo Board Porting (2019/2020)
  - Giulio Ursino - ISPU Fall Detection(2023/2024)
  - Yuri Filistad – ISPU Fall Detection (2024/2025)
- Next project: Audio Commands understanding (LLM), Advanced Equilibrium System, and more Artificial Intelligence!

life.augmented

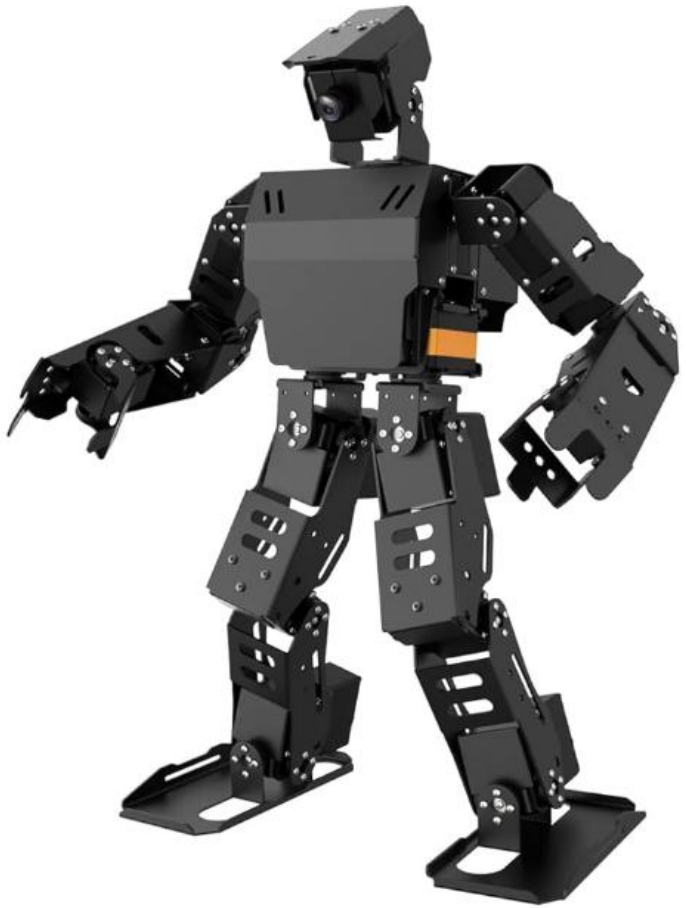
*ST Employee

Ai-Nex Robot + STM32N6



Ai-Nex Robot

- **Humanoid toy robot**
- AI-Nex robot from Hiwonder
- It allows to generate low cost PoC (to further port in full-size humanoid robotics)
- Activities:
  - MCU/MPU: STM32N6 to replace RB Pi4 (on-going)
  - ToF, Camera integration, for AI activities and SLAM, e.g.:
    - navigation (free navigation, follow-me)
    - object detection
  - ROS2 integration and Digital Twin, for real-time Avatar visualization on Display
  - New remote interface based on STEVAL-N6 DK with TouchGFX

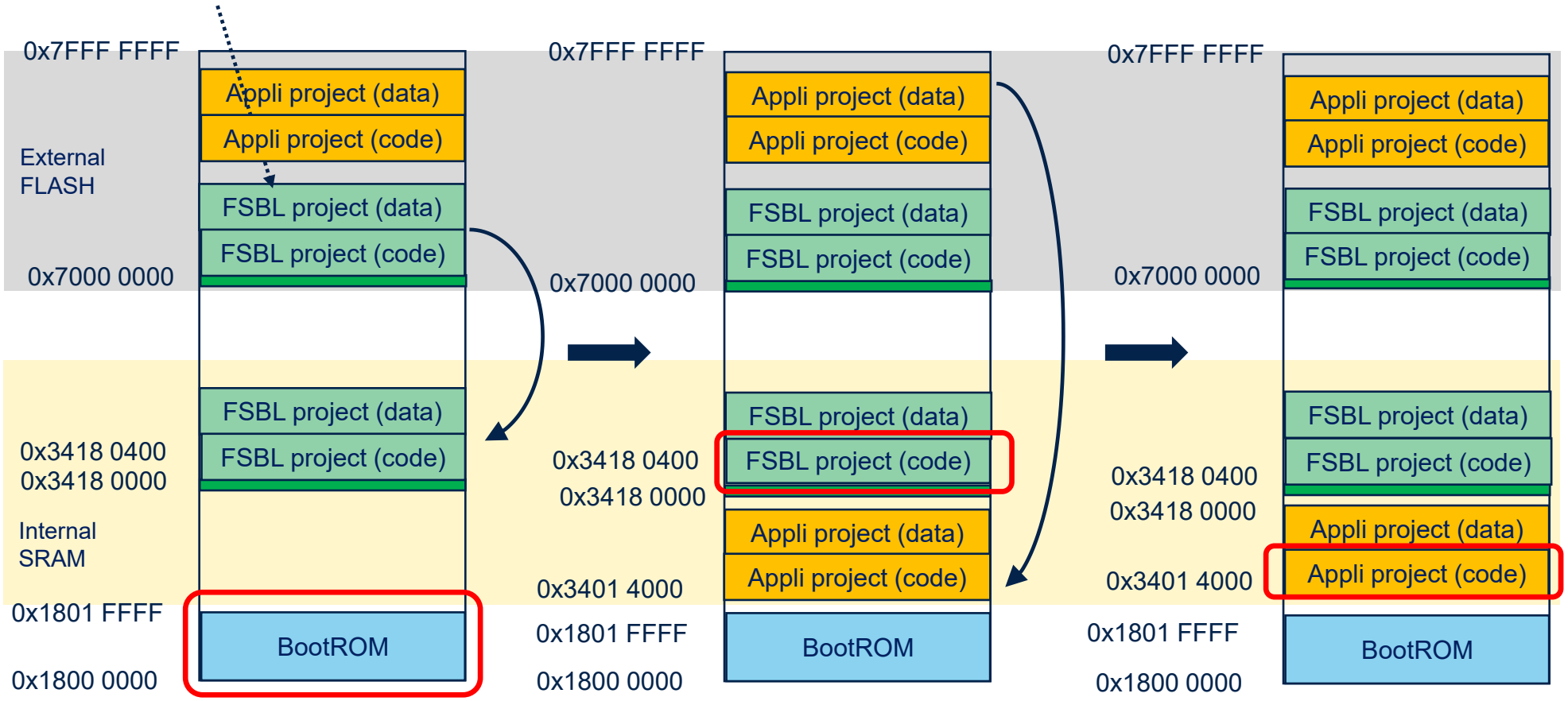


Ai-Nex Robot with ST-Nucleo

- AI-Nex Robot:
  - Raspberry based, ROS1 python code
  - Analyzed original code to identify motor protocols and AI algorithms
- Servo protocol (Lewan-Soul) implemented on STM32:
  - Implemented motor protocol on STM32F4 and tested the whole robot movements

Boot from external Flash.
FSBL + Load & Run

512-byte header (required by bootROM) + 512-byte long padding.



1) BootROM execution.
FSBL project is copied from external FLASH to internal SRAM2.
When bootROM task is done, Program Counter (PC) jumps to FSBL project first instruction location.

2) FSBL project execution.
Full appli binary (code + data) is copied from external FLASH to internal SRAM.
When FSBL project is done, PC jumps to Appli first instruction location

3) Appli execution from internal SRAM.

Remarks:
1) Appli interruptions vectors table is stored in SRAM.

2) Appli/example development is done by default in internal SRAM.
External loader needed for appli test.

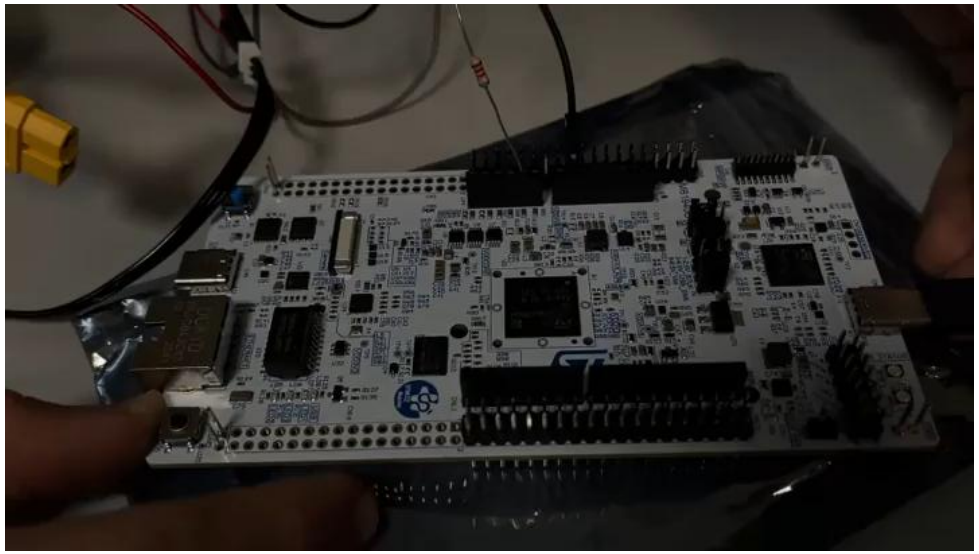
3) Application set-up: load FSBL and Appli projects in external Flash. Next, boot the device.

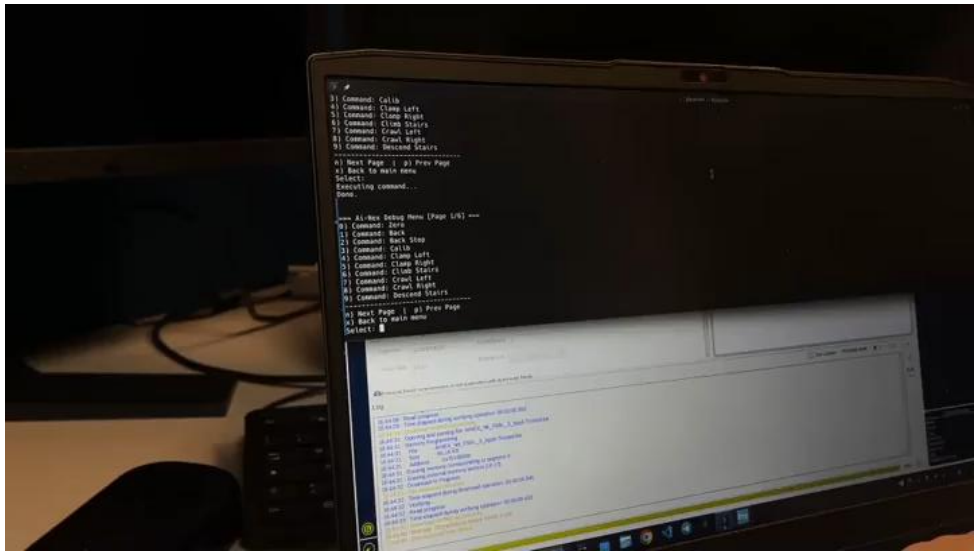
4) “smart” code to develop in SRAM1 (must be less than 80 Kbytes) to manage exit from Standby and application restart if needed. Explanation hereafter.

life.augmented

First Release on STM32N6 Nucleo

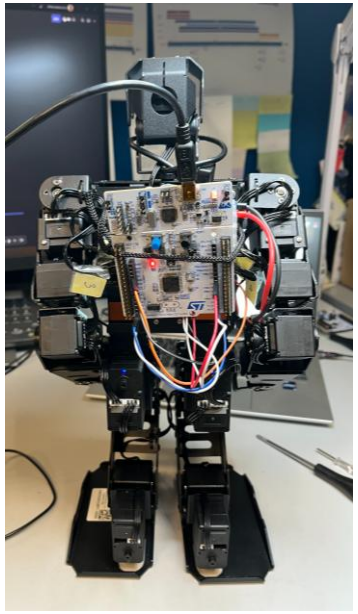
- First tests on STM32N6-Nucleo board
  - **On first video:**
    1. FSBL loaded on SRAM (green LED ON)
    2. Application loaded on SRMA (blu LED ON)
    3. Application ready (green LED OFF)
  - **On second video:**
    - The application is tested with two motors showing that the motors are moving corretly





Ai-Nex Robot with Ai-Assistant

- STM32N6 + ST camera demo:
  - SeeedStudio Ai-Assitant chosen as ST-based platform
    - **Feature 1: Flagship AI Microcontroller** Powered by the ST STM32N657, Arm® Cortex®-M55 core and integrated ST Neural-ART NPU (600 GOPS) for AI model acceleration.
    - **Feature 2: Professional-grade Vision Sensor** ST VD55G1 global shutter camera, for fast-moving objects.
    - **Feature 3: Intelligent Sensing** LSM6DSO16IS Intelligent Sensor Processing Unit (ISPU), enabling ultra-low power activity recognition without waking the main processor.
    - **Feature 4: Comprehensive Connectivity** Includes an onboard Wi-Fi module for seamless IoT connectivity, alongside a USB-C port and multiple expansion headers for maximum development flexibility.



Ai-Nex with STM32F4 Nucleo

Ai-Assitant

ST VD55G1

life.augmented

Our technology starts with You

Find out more at www.st.com

life.augmented