Robotic Surgery Project

Robotic surgery performances with Endowirst tool in UR5e robotic arm

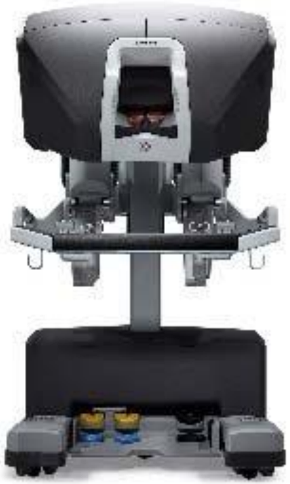
Robòtica i Control de Sistemes Biomèdics

Dr. Manel Puig i Vidal

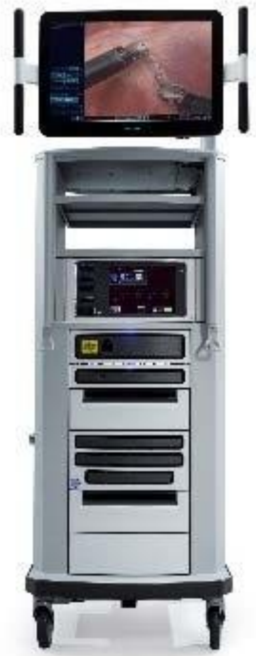
Robotic Surgery system

Main parts

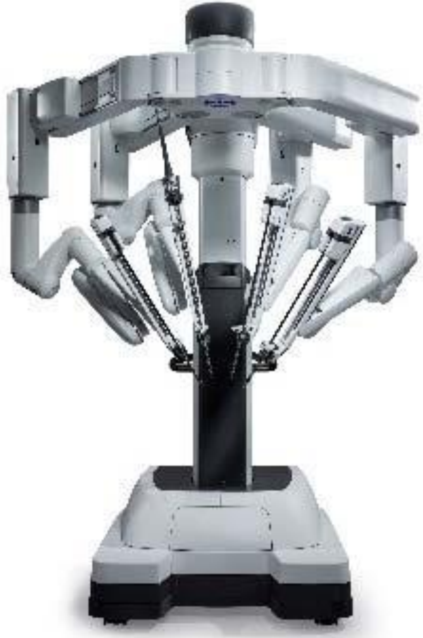
Surgeon console



Vision/control cart



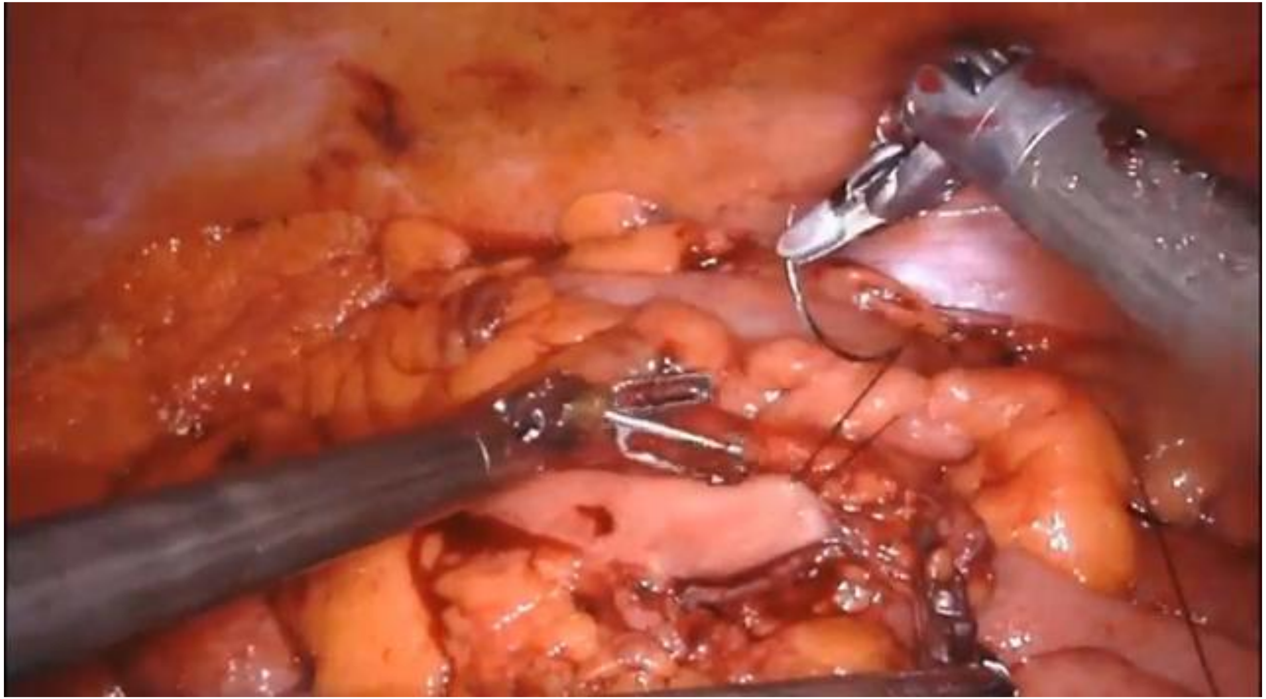
Patient-side cart



Surgery Robotics System

Real Suture process

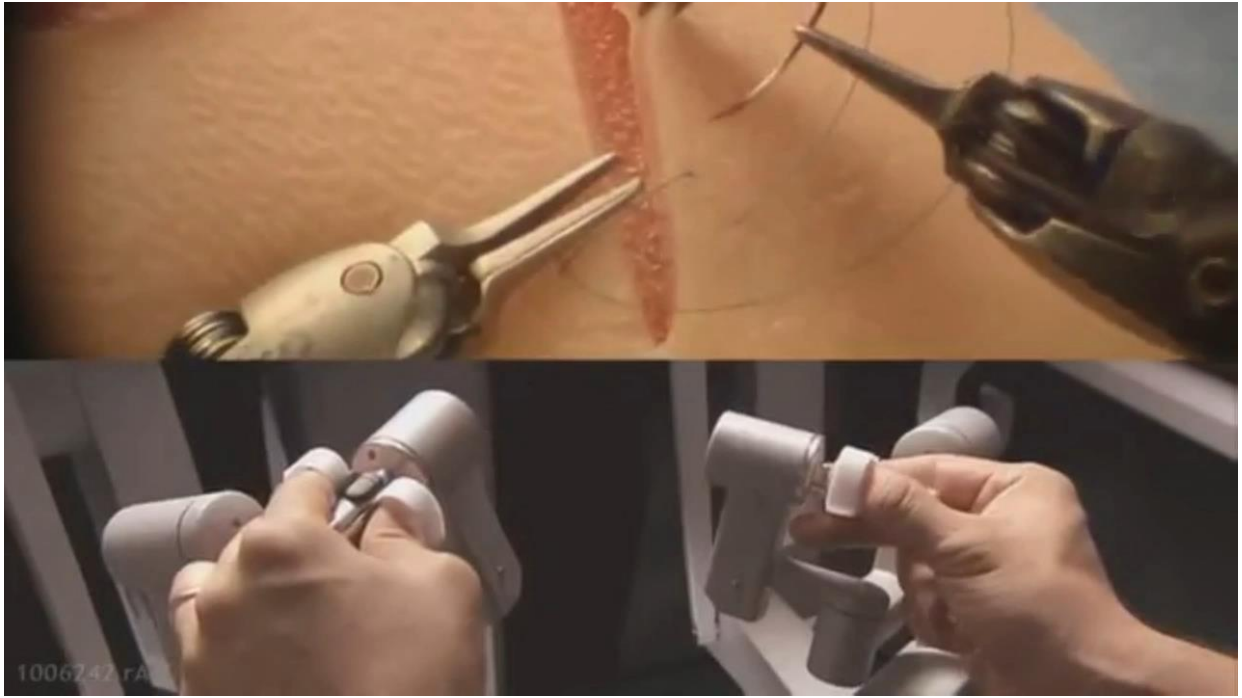
Review this suture real process

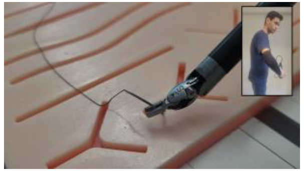


Surgery Robotics System

Training Suture process

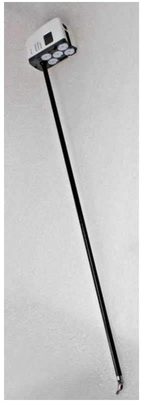
Review this training suture process

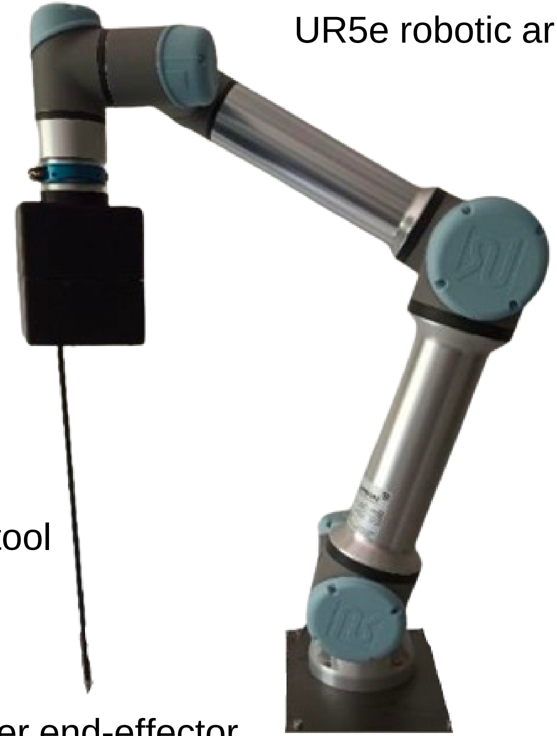




UB surgery robotics system prototype

DaVinci Xi



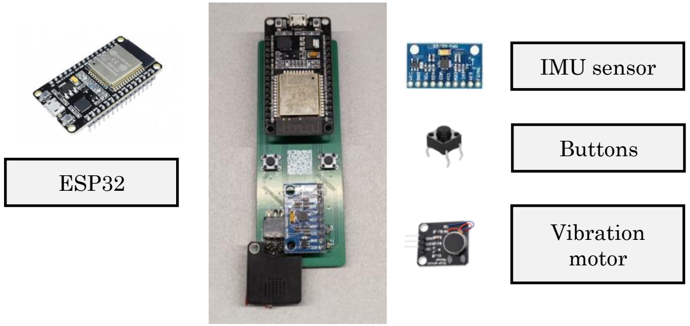


UR5e robotic arm

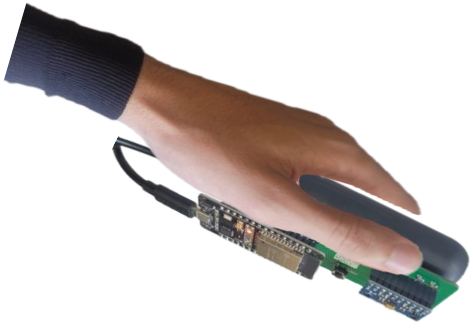
Endowrist tool

Gripper end-effector

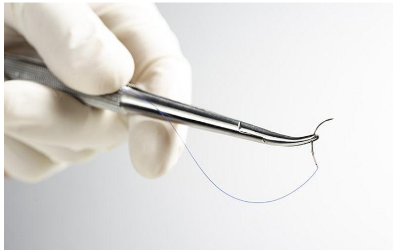
UB surgery robotics system prototype



DaVinci GUI tool

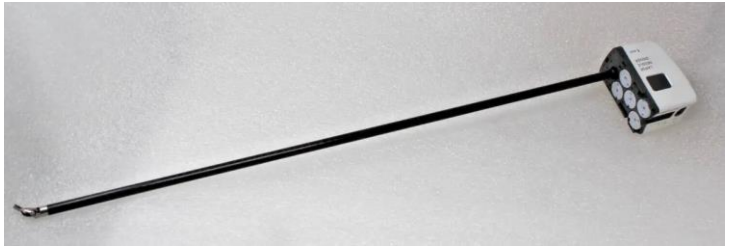


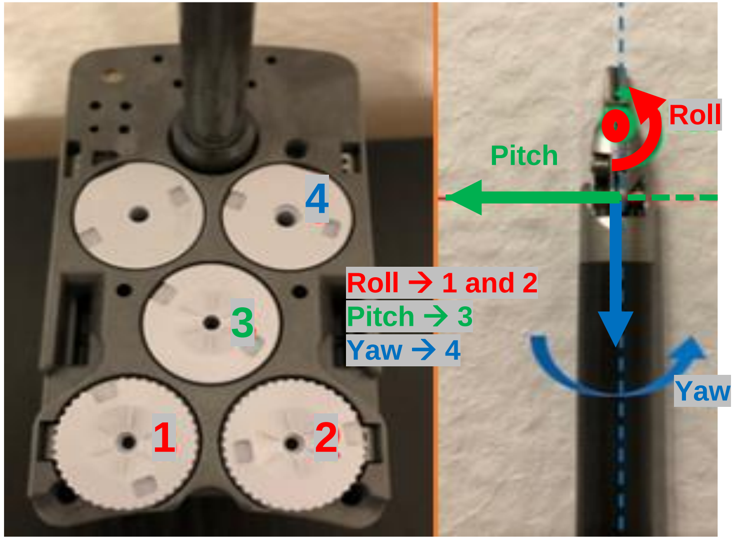
Master gripper module

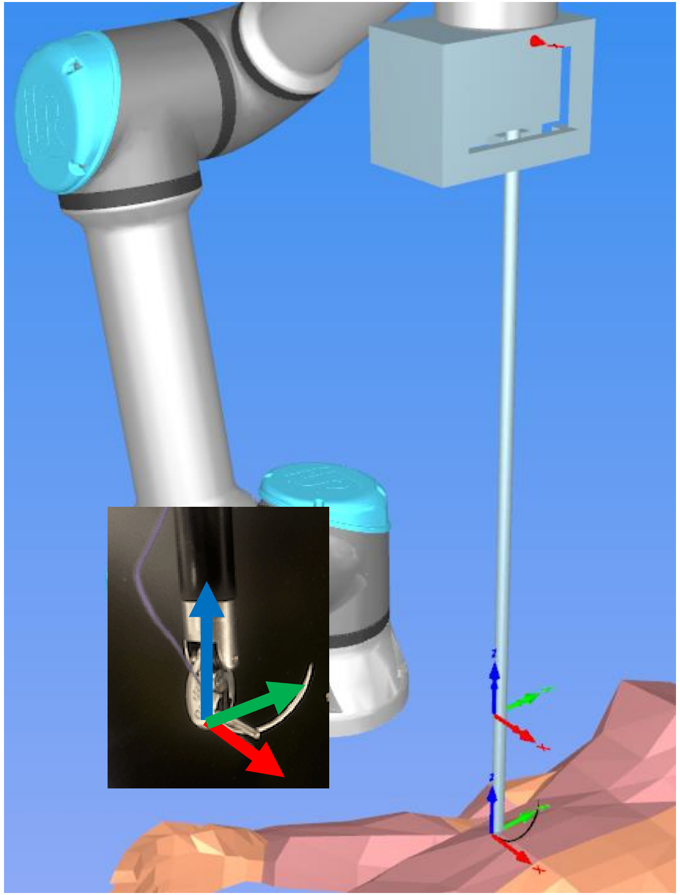


UB surgery robotics system prototype

DaVinci Endowrist Tool

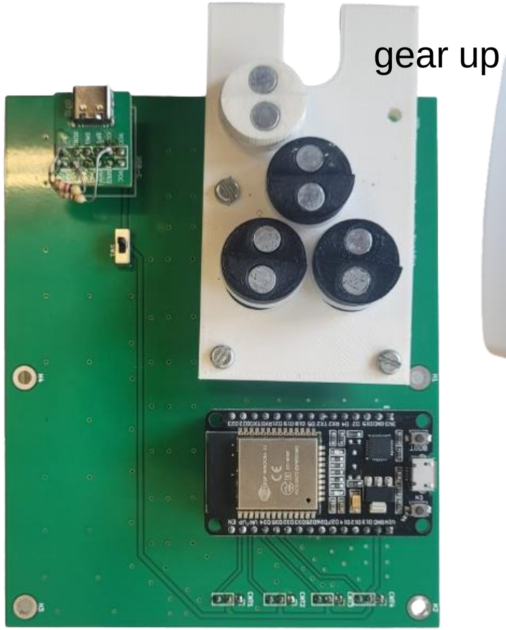




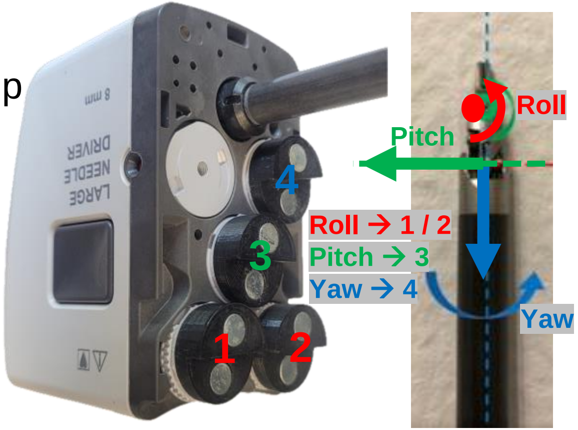


UB surgery robotics system prototype

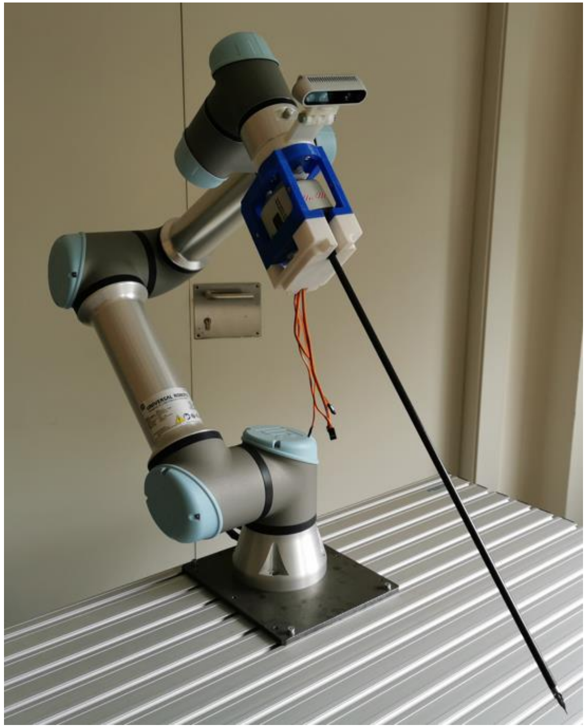
DaVinci Endowrist Tool



ESP32
(Servomotors)

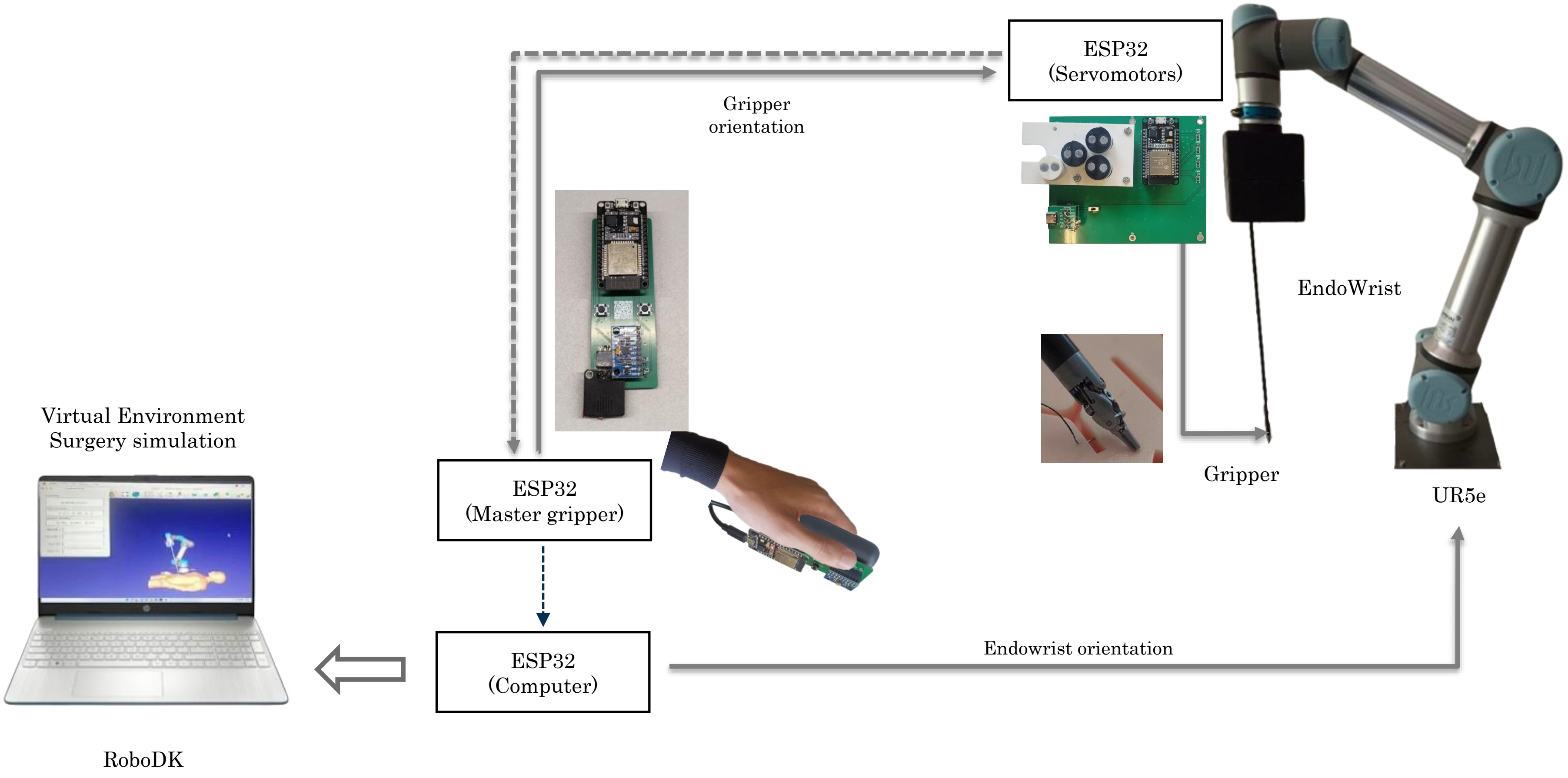


roboDK surgery program



UR5e robot arm + Endowrist tool

UB surgery robotics system prototype



Virtual Environment
Surgery simulation

RoboDK

ESP32
(Master gripper)

ESP32
(Computer)

Gripper
orientation

ESP32
(Servomotors)

EndoWrist

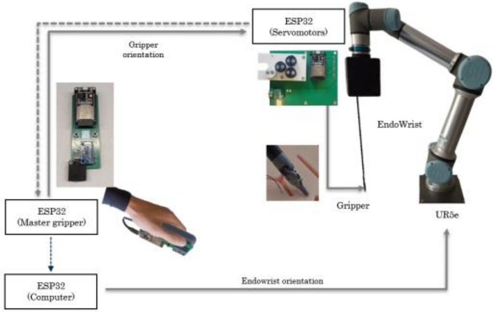
Gripper

UR5e

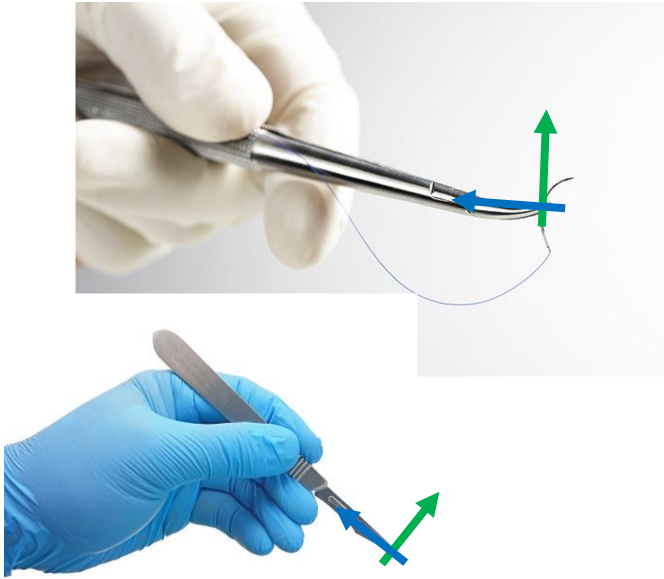
Endowrist orientation

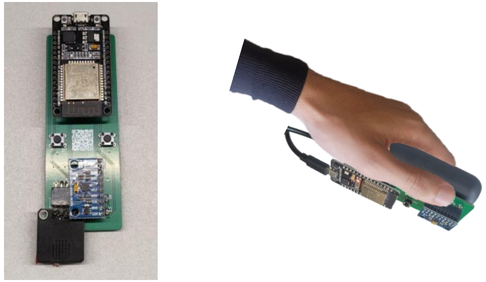
UB surgery robotics system prototype

Surgeon console



Surgeon Hand-Gripper Tool interaction ⇨ Master gripper module





- Senses the surgeon hand with IMU sensor (RPY)
- Send wireless RPY to ESP32 servomotor module

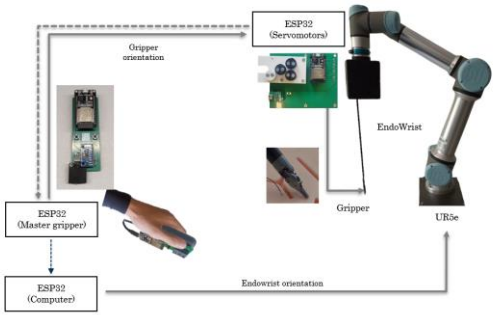
  → Gripper orientation
- Send wireless RPY to ESP32 computer module

  → Gripper/Endowrist orientation

  → RoboDK suture process simulation

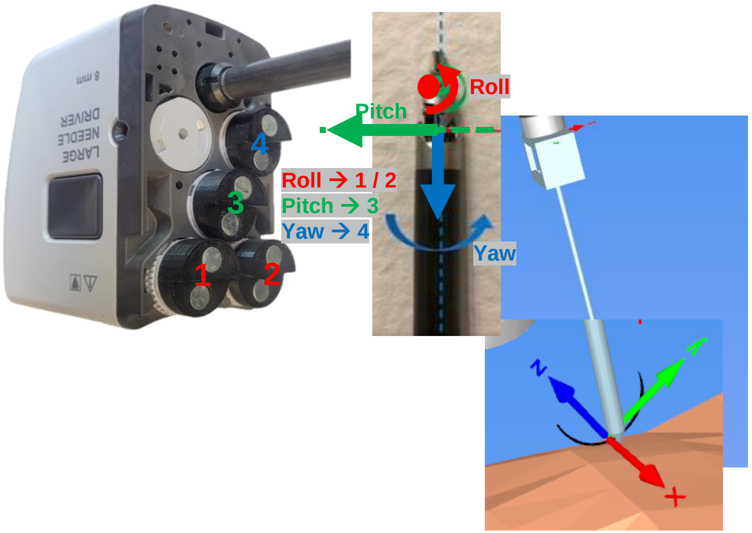
UB surgery robotics system prototype

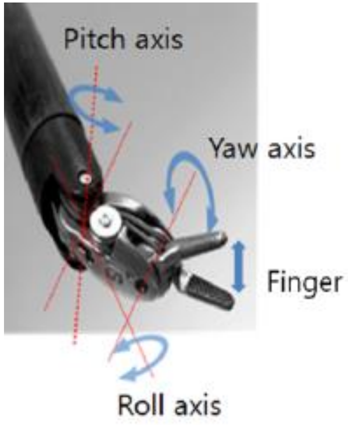
Patient-side cart

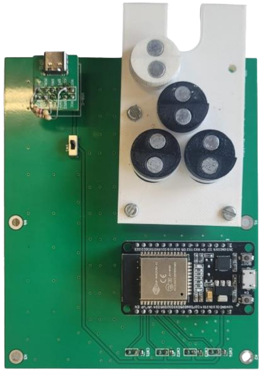


Gripper-Tool control

ESP32 servomotors module



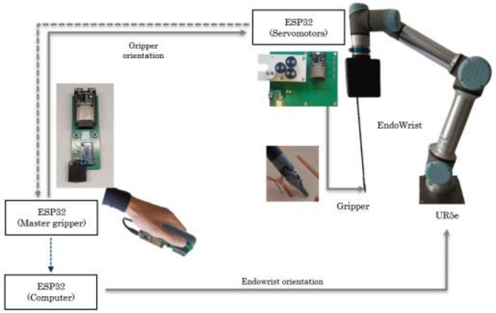




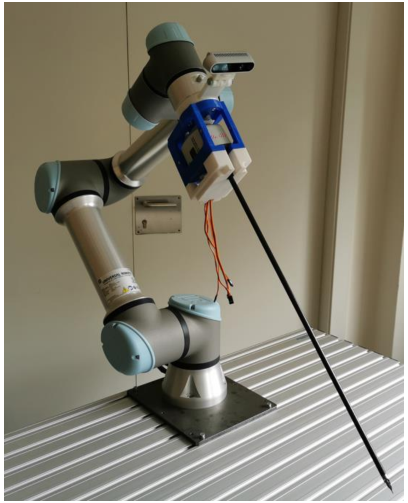
- Contains 4 servomotors (R, P, Y)
- Receives the RPY angles from ESP32 Master gripper module
- Drives each servomotor → RPY Orientation
- Reads the servomotor current → torque

Vision/control cart

UB surgery robotics system prototype



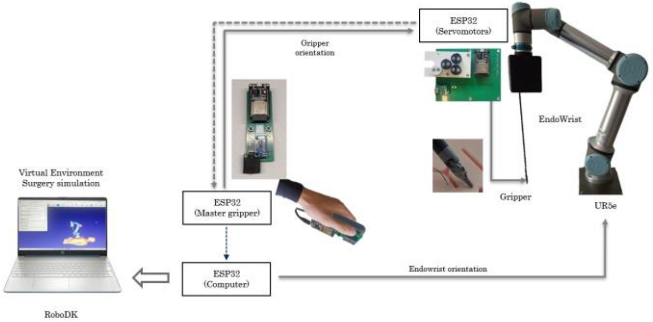
Communication and control ⇨ ESP32 Computer module



- Receives all information from ESP32 Master module:
  - the RPY angles
  - Push buttons for mode operation
  - The torques
- Performs offline simulation suture process
- Performs Endowrist tool orientation

UB surgery robotics system prototype

Vision/control cart



ESP32
(Computer)

- USB serial Data reading
- UR5e Endowrist orientation
- Gripper orientation
- Torque values

- RoboDK simulation

Python

- Real-time control:
  UR5e+Endowrist
  Gripper

- Ethernet

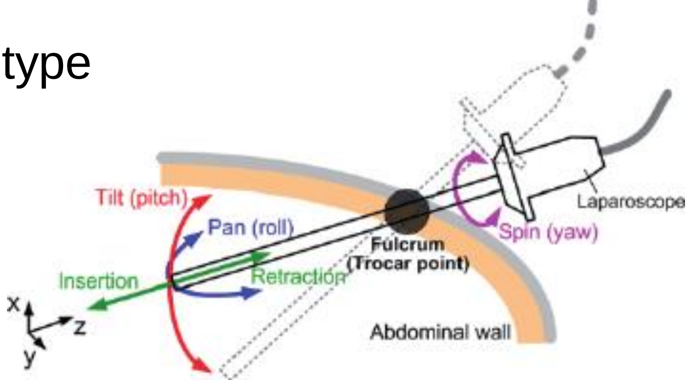
- Sockets communication

UB surgery robotics system prototype

Modes of operation

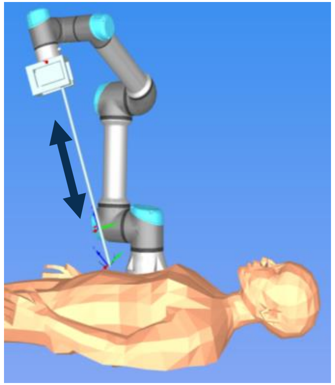
Endowrist mode: ⇨ Endowrist orientation at Fulcrum point

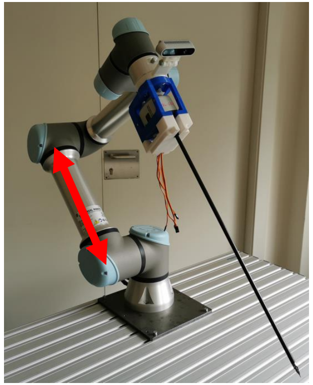
Push “e” key from keyboard



Insertion/retraction mode: ⇨ Gripper Up/Down

Push “u”or “d” key from keyboard





Gripper mode: ⇨

Gripper orientation

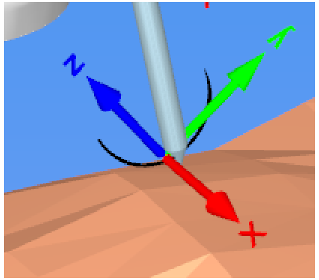
Push s1 and s2 buttons from ESP32 Master gripper

Open Gripper

Push s1 button from ESP32 Master gripper

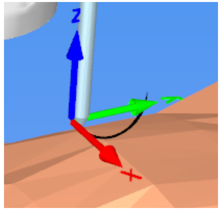
Close Gripper

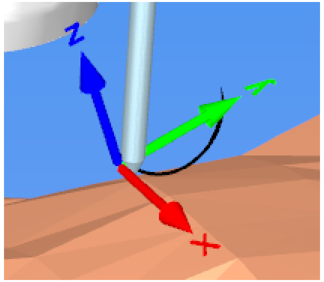
Push s2 button from ESP32 Master gripper

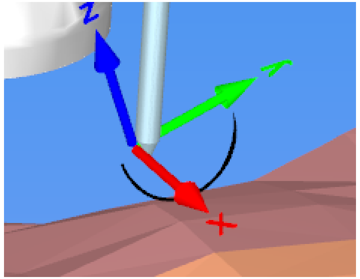


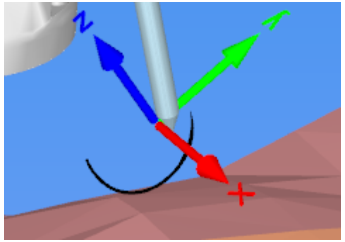
Proposed Suture process:

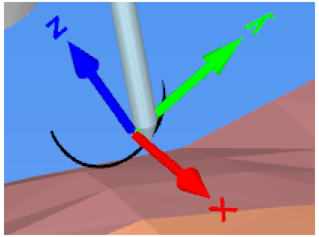
1. Close gripper
2. Rotx gripper
3. Rotx Endowrist
4. Open gripper
5. Endowrist up
6. Rotx Endowrist
7. Endowrist down
8. Close gripper
9. Rotx gripper
10. Rotx Endowrist

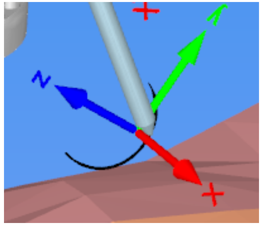












Robotic Surgery Project objectives:

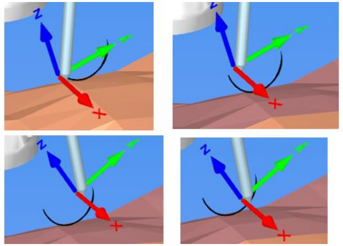
Session 1:

Previous Task:

Proposed roboDK suture process (**A3**)

Lab session:

Review system performances and practice the proposed suture process



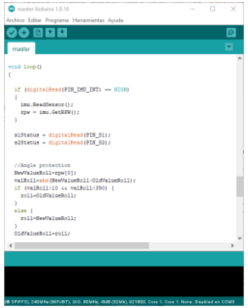
Session 2:

Previous task:

Actual suture limitations and proposed HW-Arduino improvements (**D2.2P**)

Lab session:

Implement HW-Arduino proposed improvements



```
void loop()
{

  if (digitalRead(PIN_IMU_INT) == HIGH)
  {
    imu.ReadSensor();
    rpw = imu.GetRPW();
  }

  s1Status = digitalRead(PIN_S1);
  s2Status = digitalRead(PIN_S2);


  //Angle protection
  NewValueRoll=rpw[0];
  valRoll=abs(NewValueRoll-OldValueRoll);
  if (valRoll>10 && valRoll<350) {
    roll=OldValueRoll;
  }
  else {
    roll=NewValueRoll;
  }
  OldValueRoll=roll;
```

```
B SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, Core 1, Core 1, None, Disabled en COM3
```

Session 3:

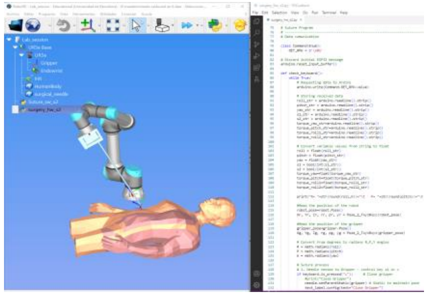
Previous Task:

proposed SW-roboDK improvements (**D2.3P**)

Lab session:

Implement proposed SW-roboDK improvements

Connection with HW-Arduino proposed improvements



```
# Suture Program
# ----------------------------------------
# Data communication

class Command(Enum):
    [illegible]

# [illegible]
arduino.reset_input_buffer()

def check_keyboard():
    while True:
        # [illegible]
        [illegible]

        # Storing received data
        [illegible]

        # Convert [illegible]
        [illegible]

        [illegible]

        # Suture process
        [illegible]
```

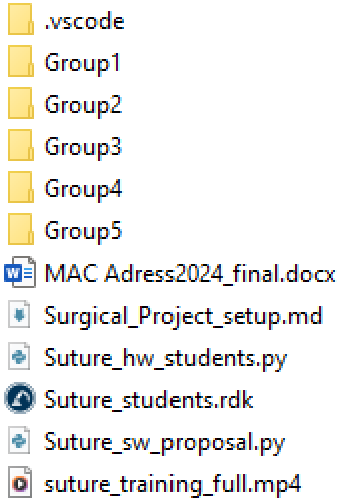
Session 4:

Validation

Project presentation

Session 1:

Open “Surgery_students2024” Project folder in VS code



.vscode

Group1

Group2

Group3

Group4

Group5

MAC Adress2024_final.docx

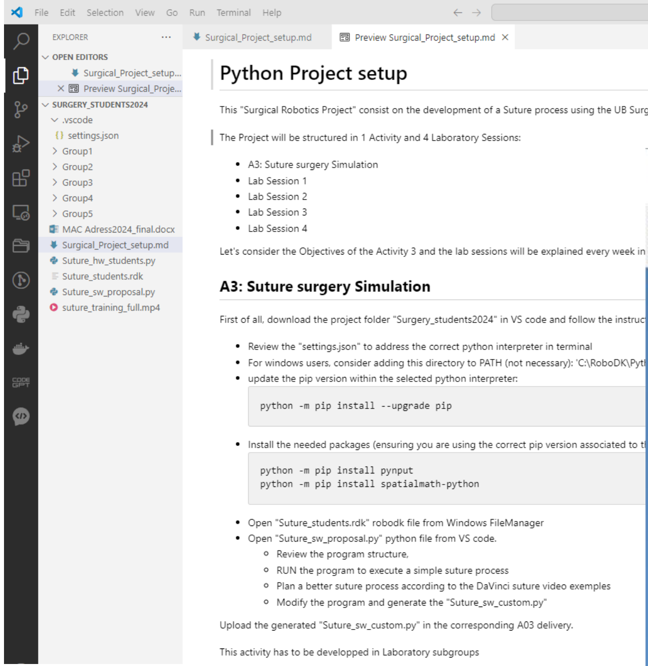
Surgical_Project_setup.md

Suture_hw_students.py

Suture_students.rdk

Suture_sw_proposal.py

suture_training_full.mp4


File
Edit
Selection
View
Go
Run
Terminal
Help
Surgery_students2024
EXPLORER
OPEN EDITORS
Surgical_Project_setup...
Preview Surgical_Proje...
Surgical_Project_setup.md
Preview Surgical_Project_setup.md

SURGERY_STUDENTS2024

- .vscode
  - settings.json
- Group1
- Group2
- Group3
- Group4
- Group5
- MAC Adress2024_final.docx
- Surgical_Project_setup.md
- Suture_hw_students.py
- Suture_students.rdk
- Suture_sw_proposal.py
- suture_training_full.mp4

Python Project setup

This "Surgical Robotics Project" consist on the development of a Suture process using the UB Surgical Robotic System.

The Project will be structured in 1 Activity and 4 Laboratory Sessions:

- A3: Suture surgery Simulation
- Lab Session 1
- Lab Session 2
- Lab Session 3
- Lab Session 4

Let's consider the Objectives of the Activity 3 and the lab sessions will be explained every week in

A3: Suture surgery Simulation

First of all, download the project folder "Surgery_students2024" in VS code and follow the instruc

- Review the "settings.json" to address the correct python interpreter in terminal
- For windows users, consider adding this directory to PATH (not necessary): 'C:\RoboDK\Pyt
- update the pip version within the selected python interpreter:

```
python -m pip install --upgrade pip
```

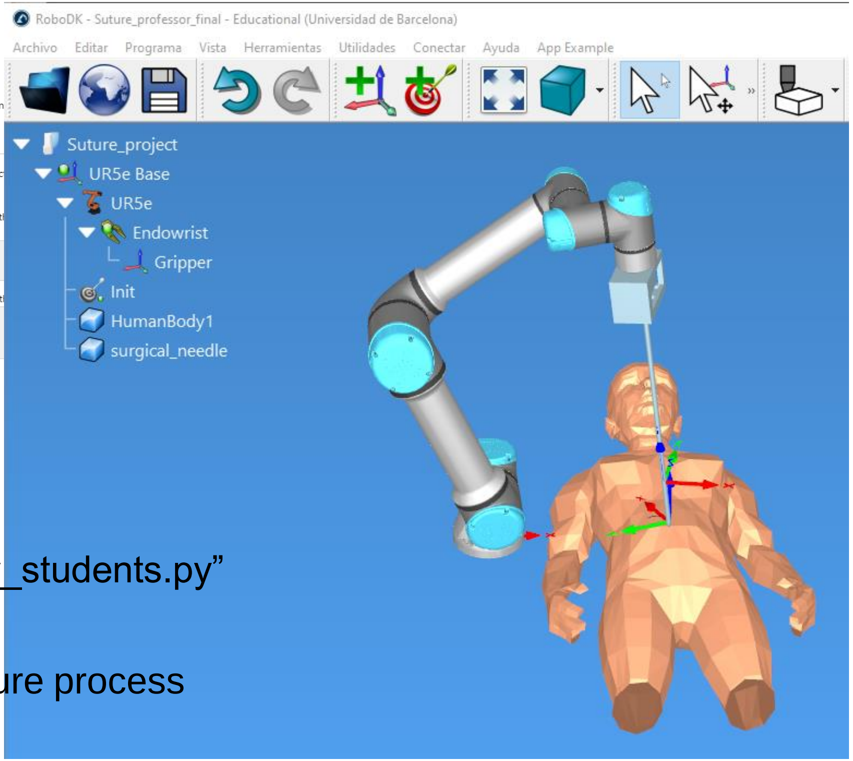
- Install the needed packages (ensuring you are using the correct pip version associated to t

```
python -m pip install pynput
python -m pip install spatialmath-python
```

- Open "Suture_students.rdk" robodk file from Windows FileManager
- Open "Suture_sw_proposal.py" python file from VS code.
  - Review the program structure,
  - RUN the program to execute a simple suture process
  - Plan a better suture process according to the DaVinci suture video exemples
  - Modify the program and generate the "Suture_sw_custom.py"

Upload the generated "Suture_sw_custom.py" in the corresponding A03 delivery.

This activity has to be developped in Laboratory subgroups

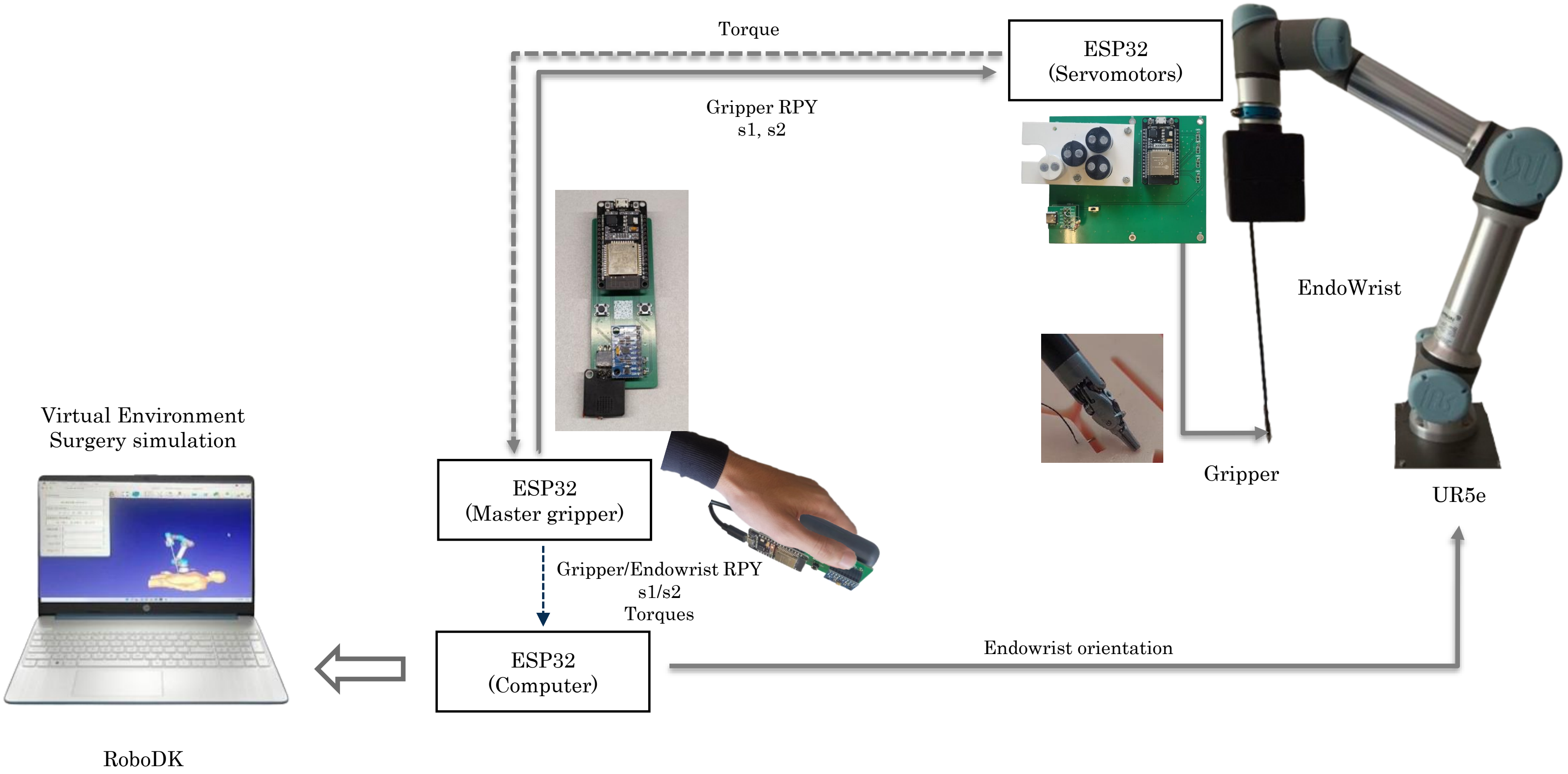


1. Connect:
ESP32_computer
ESP32_master
ESP32_servos

2. Open Project and run “Suture_hw_students.py”

3. Try to execute your proposed suture process

UB surgery robotics system prototype



Torque

ESP32
(Servomotors)

Gripper RPY
s1, s2

EndoWrist

Virtual Environment
Surgery simulation

ESP32
(Master gripper)

Gripper

UR5e

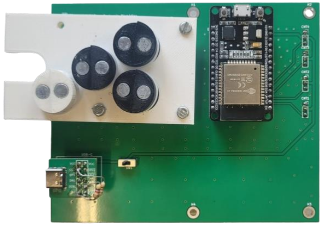
Gripper/Endowrist RPY
s1/s2
Torques

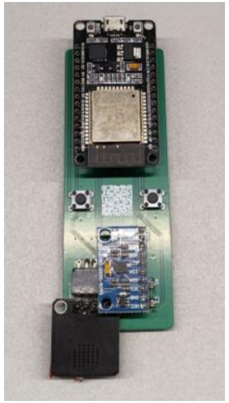
ESP32
(Computer)

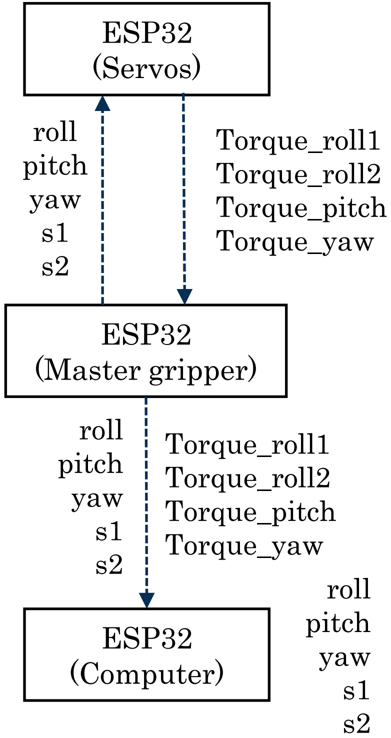
Endowrist orientation

RoboDK

Detailed UB surgical robotic system







ESP32
(Servos)

roll
pitch
yaw
s1
s2

Torque_roll1
Torque_roll2
Torque_pitch
Torque_yaw

ESP32
(Master gripper)

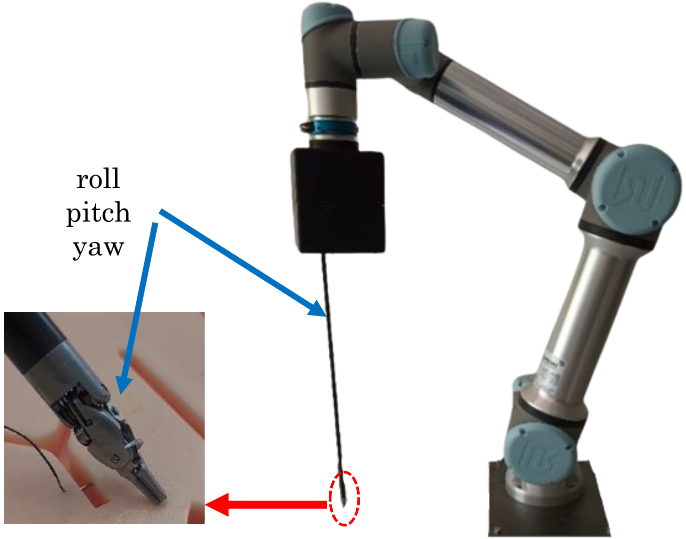
roll
pitch
yaw
s1
s2

Torque_roll1
Torque_roll2
Torque_pitch
Torque_yaw

ESP32
(Computer)

roll
pitch
yaw
s1
s2

Torque_yaw
Torque_pitch
Torque_roll1
Torque_roll2

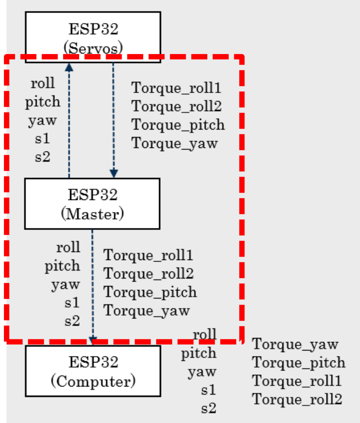


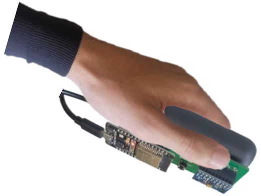
roll
pitch
yaw

ESP32
(Master gripper)

Data Structure definitions

```
#include "src/RoboticsUB.h"
#include <esp_now.h>
#include <WiFi.h>
…
// Esta es la estructura de los datos que enviaremos del master a los servomotores
typedef struct {
    float roll;
    float pitch;
    float yaw;
    int s1Status;
    int s2Status;
} TxMessage;
// Creamos una varaible con la estructura recien creada
TxMessage dataToServos;
// Esta es la estructura de los datos que enviaremos del master al computer
typedef struct {
    float roll;
    float pitch;
    float yaw;
    int s1Status;
    int s2Status;
    float torque_yaw;
    float torque_pitch;
    float torque_roll1;
    float torque_roll2;
} Tx2Message;
// Creamos una varible con la estructura recien creada
Tx2Message dataToComputer;
// Esta es la estructura de los datos que reciviremos
typedef struct {
    float torque_yaw;
    float torque_pitch;
    float torque_roll1;
    float torque_roll2;
} RxMessage;
// Creamos una varaible con la estructura recien creada
RxMessage dataFromServos;
…
```





ESP32
(Master 1)

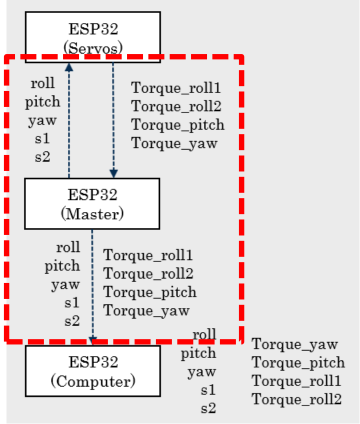
ESP32
(Master gripper)

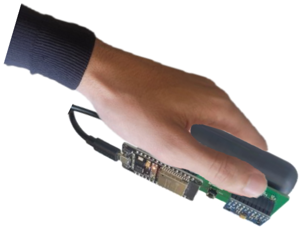
```
void loop() {
  if (digitalRead(PIN_IMU_INT) == HIGH)
  {
    imu.ReadSensor();
    rpw = imu.GetRPW();
  }
  s1Status = digitalRead(PIN_S1);
  s2Status = digitalRead(PIN_S2);
 //Angle protection
  NewValueRoll=rpw[0];
  valRoll=abs(NewValueRoll-OldValueRoll);
  if (valRoll>10 && valRoll<350) {
    roll=OldValueRoll;
  }
  else {
    roll=NewValueRoll;
  }
  OldValueRoll=roll;
…
  //Enviar los datos a los servomotores
  dataToServos.roll=roll;
  dataToServos.pitch=pitch;
  dataToServos.yaw=fmod(yaw + zero_yaw, 360.0);//New
  dataToServos.s1Status=s1Status;
  dataToServos.s2Status=s2Status;
  //Recibir datos de servomotor y enviarlos al computer
  dataToComputer.torque_yaw=dataFromServos.torque_yaw;
  dataToComputer.torque_pitch=dataFromServos.torque_pitch;
  dataToComputer.torque_roll1=dataFromServos.torque_roll1;
  dataToComputer.torque_roll2=dataFromServos.torque_roll2;
  //Enviar los datos al computer
  dataToComputer.roll=roll;
  dataToComputer.pitch=pitch;
  dataToComputer.yaw=fmod(yaw + zero_yaw, 360.0);//New
  dataToComputer.s1Status=s1Status;
  dataToComputer.s2Status=s2Status;
…
}
```

Read IMU rpy
Read s1 and s2

Filter angle values > 10deg

Receive/ Send Data


ESP32 (Servos)
roll pitch yaw s1 s2
Torque_roll1 Torque_roll2 Torque_pitch Torque_yaw
ESP32 (Master)
roll pitch yaw s1 s2
Torque_roll1 Torque_roll2 Torque_pitch Torque_yaw
ESP32 (Computer)
roll pitch yaw s1 s2
Torque_yaw Torque_pitch Torque_roll1 Torque_roll2

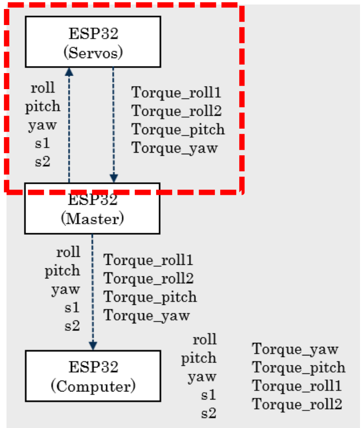


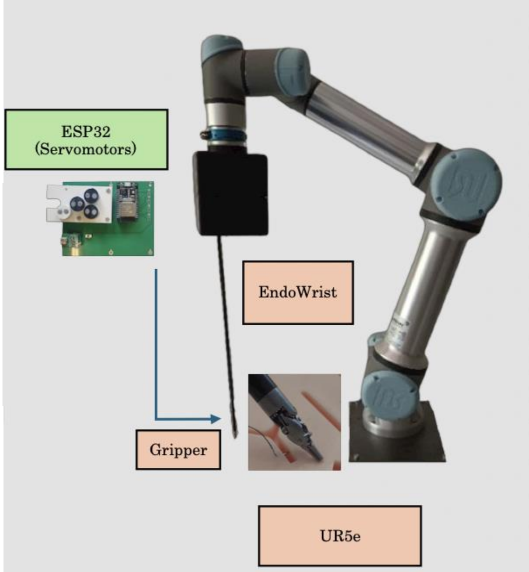
ESP32
(Servos)

```
#include <esp_now.h>
#include <WiFi.h>
#include "src/RoboticsUB.h"
#include <ESP32Servo.h>
// Direccion MAC del master
uint8_t masterMacAddress[] = {0x0c, 0xb8, 0x15, 0xd7, 0xe1, 0x7c};
…
//estructura de datos que enviara este al master
typedef struct {
  float torque_yaw;
  float torque_pitch;
  float torque_roll1;
  float torque_roll2;
} TxMessage;
//variable del tipo estructura para enviar datos
TxMessage dataToMaster;
//estructura de datos que recibiremos del master
typedef struct {
  float roll;
  float pitch;
  float yaw;
  int s1Status;
  int s2Status;
} RxMessage;
//variable del tipo estructura para recibir datos
RxMessage dataFromMaster;
void setup() {
…
}
void loop() {
…
//Enviamos los valores del torque al Master
  dataToMaster.torque_yaw=torque_yaw;
  dataToMaster.torque_pitch=torque_pitch;
  dataToMaster.torque_roll1=torque_roll1;
  dataToMaster.torque_roll2=torque_roll2;
}
```

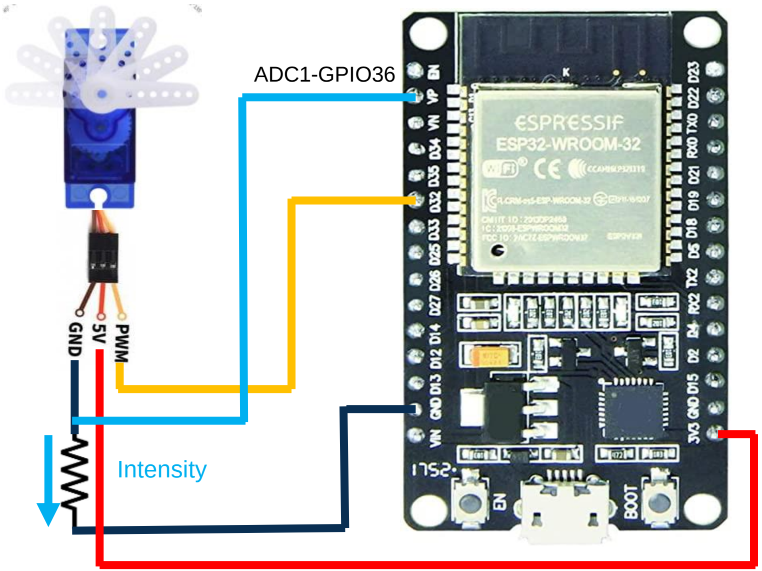
Data Structure definitions

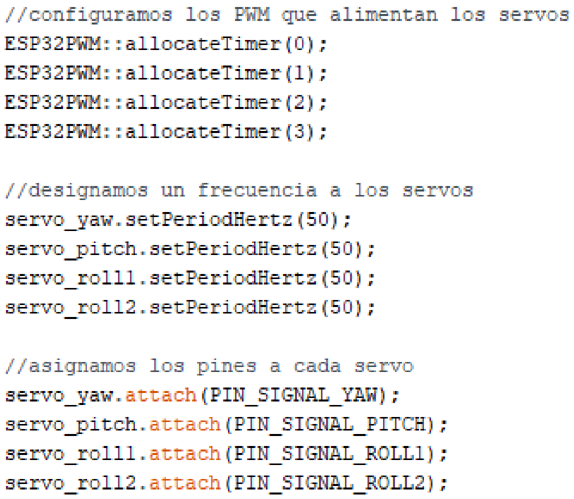
Receive/ Send Data



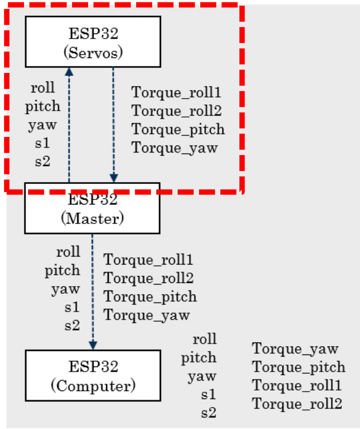


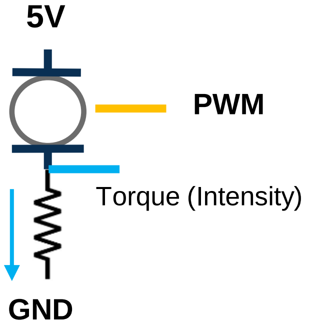
SG90 servomotor:

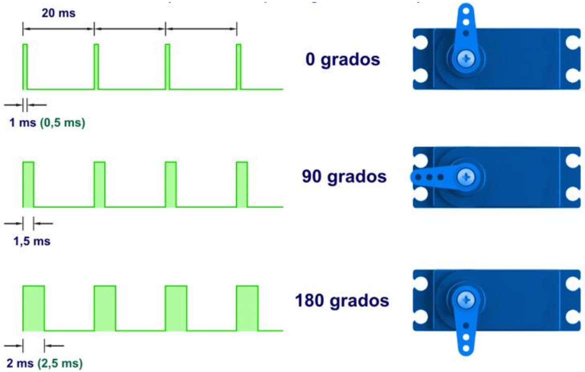




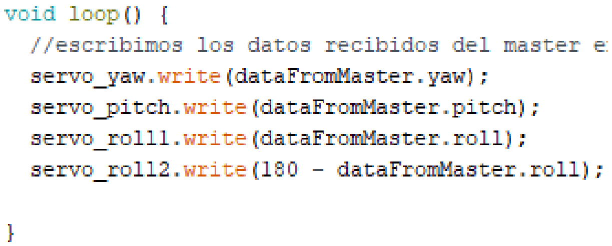
```
//configuramos los PWM que alimentan los servos
ESP32PWM::allocateTimer(0);
ESP32PWM::allocateTimer(1);
ESP32PWM::allocateTimer(2);
ESP32PWM::allocateTimer(3);

//designamos un frecuencia a los servos
servo_yaw.setPeriodHertz(50);
servo_pitch.setPeriodHertz(50);
servo_roll1.setPeriodHertz(50);
servo_roll2.setPeriodHertz(50);

//asignamos los pines a cada servo
servo_yaw.attach(PIN_SIGNAL_YAW);
servo_pitch.attach(PIN_SIGNAL_PITCH);
servo_roll1.attach(PIN_SIGNAL_ROLL1);
servo_roll2.attach(PIN_SIGNAL_ROLL2);
```





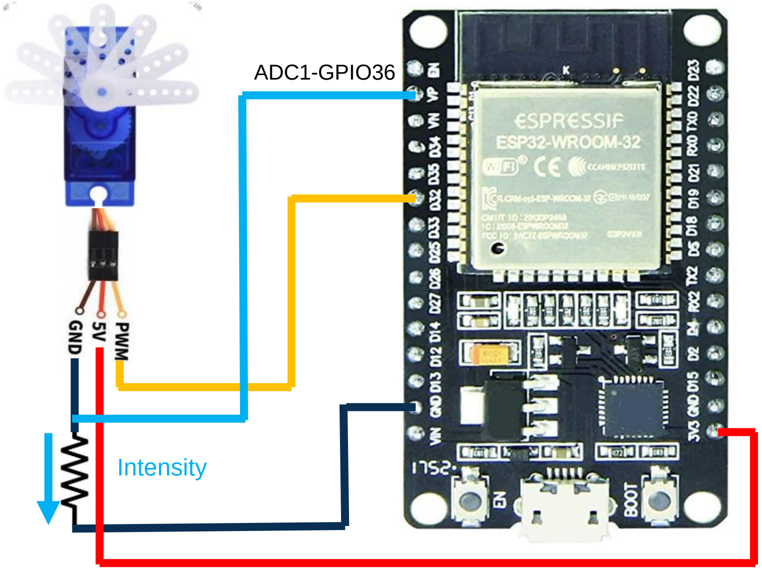


angle application:

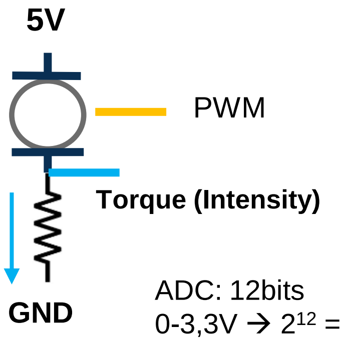


```
void loop() {
  //escribimos los datos recibidos del master e
  servo_yaw.write(dataFromMaster.yaw);
  servo_pitch.write(dataFromMaster.pitch);
  servo_roll1.write(dataFromMaster.roll);
  servo_roll2.write(180 - dataFromMaster.roll);

}
```

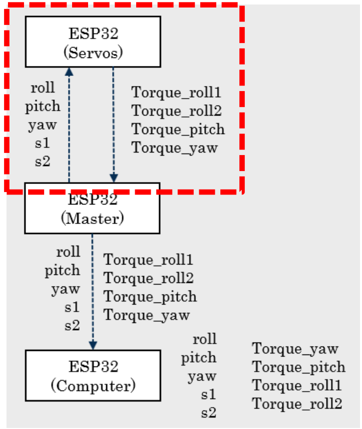
SG90 servomotor:



Torque:

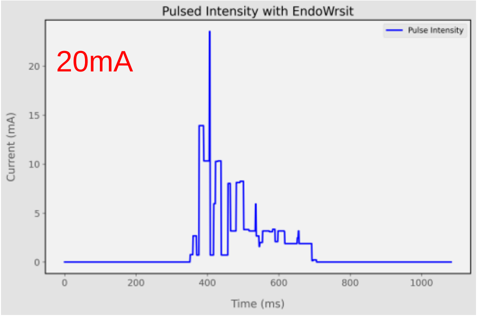


ADC: 12bits
0-3,3V → 2^{12} =4096 digital states

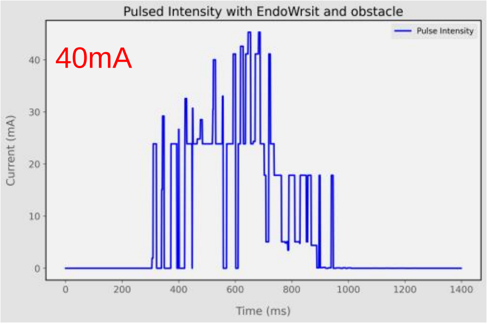


$$I_{yaw} = \frac{V_{ADC1}\frac{3,3}{4096}}{R_{shunt}}$$

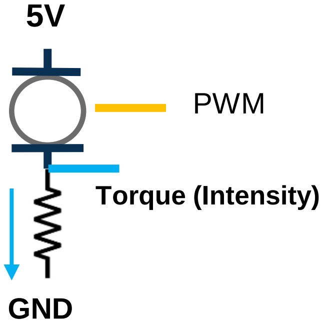
Without obstacle



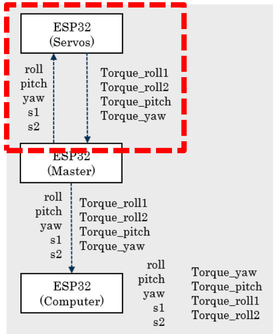
With obstacle



SG90 servomotor:



Torque:


ESP32 (Servos)

roll
pitch
yaw
s1
s2

Torque_roll1
Torque_roll2
Torque_pitch
Torque_yaw

ESP32 (Master)

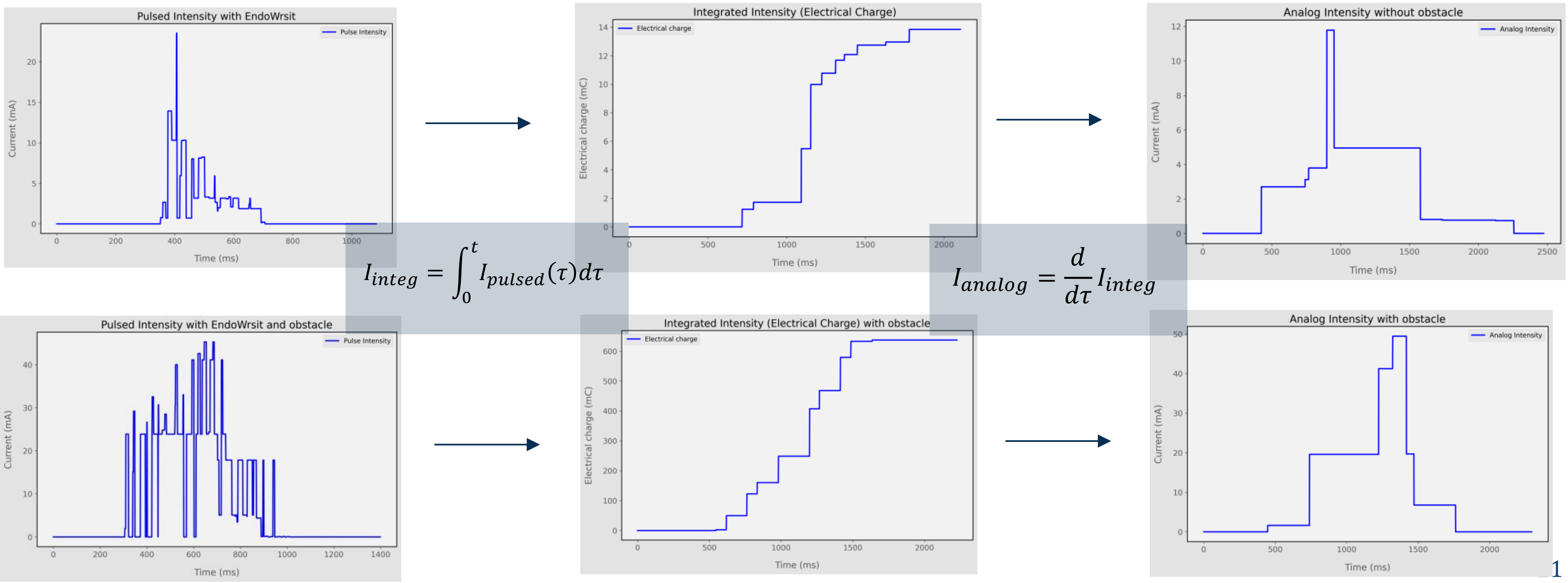
roll
pitch
yaw
s1
s2

Torque_roll1
Torque_roll2
Torque_pitch
Torque_yaw

ESP32 (Computer)

roll
pitch
yaw
s1
s2

Torque_yaw
Torque_pitch
Torque_roll1
Torque_roll2

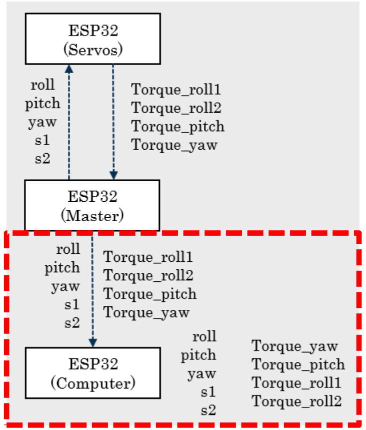


$$I_{integ} = \int_0^t I_{pulsed}(\tau)d\tau$$

$$I_{analog} = \frac{d}{d\tau} I_{integ}$$

ESP32
(Computer)

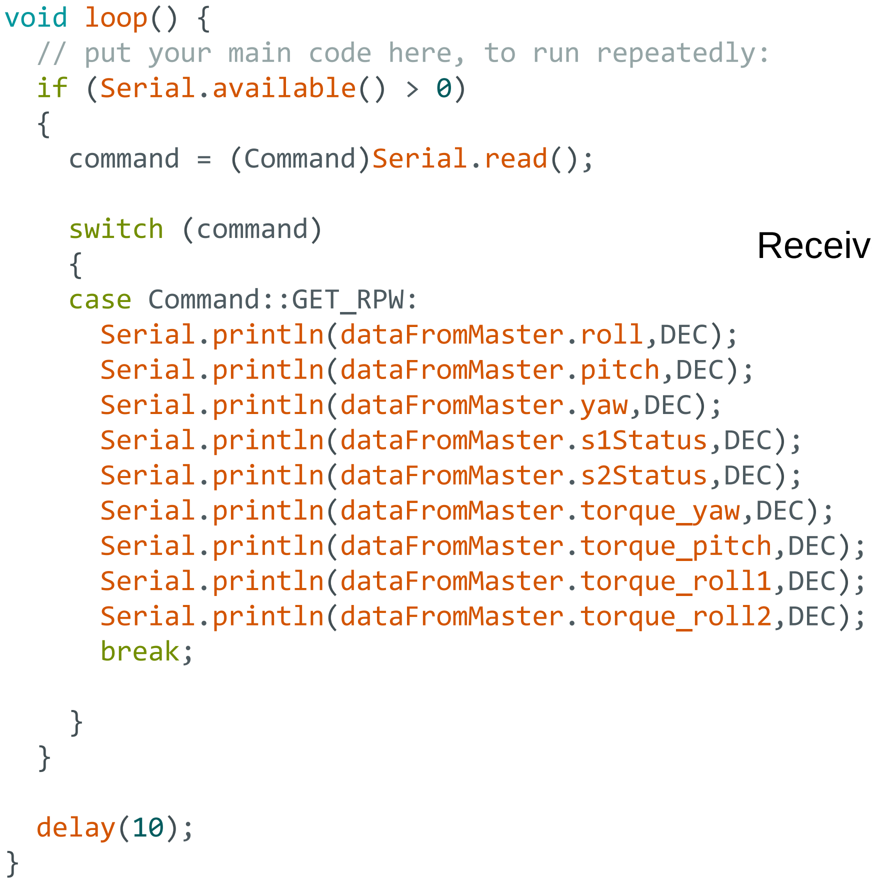
```
#include <esp_now.h>
#include <WiFi.h>
#include "src/RoboticsUB.h"
#include <ESP32Servo.h>
enum class Command : byte
{
  GET_RPW = 1
};
Command command = Command::GET_RPW;
// Direccion MAC del master
uint8_t masterMacAddress[] = {0x0c, 0xb8, 0x15, 0xd7, 0xe1, 0x7c};
// Esta es la estructura de los datos que reciviremos
typedef struct {
    float roll;
    float pitch;
    float yaw;
    int s1Status;
    int s2Status;
    float torque_yaw;
    float torque_pitch;
    float torque_roll1;
    float torque_roll2;
} RxMessage;
// Creamos una varaible con la estructura recien creada
RxMessage dataFromMaster;
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  // Copiamos los datos recibidos a nuestra variable dataFromMaster
  memcpy(&dataFromMaster, incomingData, sizeof(dataFromMaster));

}
void setup() {
…
}
```



Data Structure definitions

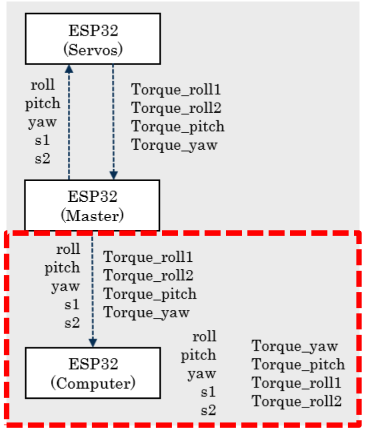
ESP32
(Computer)

ESP32
(Computer)



```
void loop() {
  // put your main code here, to run repeatedly:
  if (Serial.available() > 0)
  {
    command = (Command)Serial.read();

    switch (command)
    {
    case Command::GET_RPW:
      Serial.println(dataFromMaster.roll,DEC);
      Serial.println(dataFromMaster.pitch,DEC);
      Serial.println(dataFromMaster.yaw,DEC);
      Serial.println(dataFromMaster.s1Status,DEC);
      Serial.println(dataFromMaster.s2Status,DEC);
      Serial.println(dataFromMaster.torque_yaw,DEC);
      Serial.println(dataFromMaster.torque_pitch,DEC);
      Serial.println(dataFromMaster.torque_roll1,DEC);
      Serial.println(dataFromMaster.torque_roll2,DEC);
      break;

    }
  }

  delay(10);
}
```

Receive/ Send Data



ESP32
(Computer)

ESP32
(Computer)

Python

- RoboDK simulation

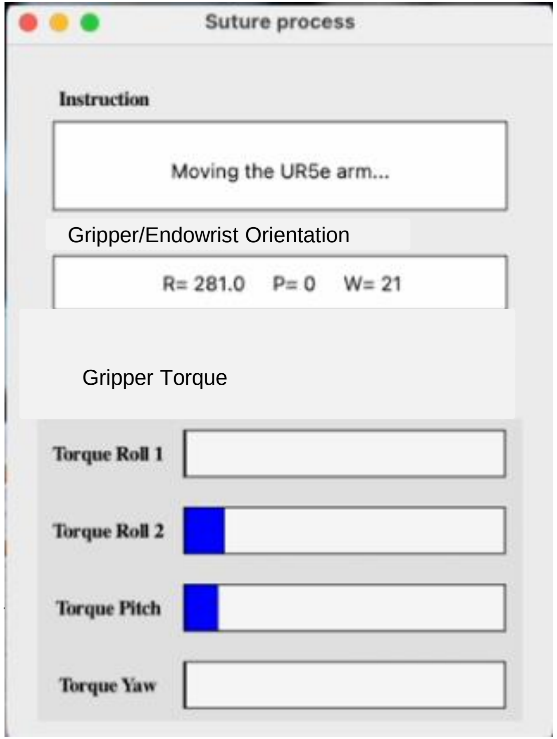
- USB serial Data reading
- UR5e Endowrist orientation
- Gripper orientation
- RPY and Torque values

- Real-time control:
  UR5e+Endowrist
  Gripper

- Ethernet
- Sockets communication

RoboDK python simulation program

```
# Import Libraries
# Constants
# UR5e connection online with roboDK
# Initialize Arduino
def initialize_arduino():
# Initialize RoboDK
def initialize_robodk():
# Handle suture process
def suture(arduino, robot, gripper, needle, base, text_labe
# Main function
def main():
    arduino = initialize_arduino()
    RDK, robot, base, endowrist, gripper, needle, Init_target = initialize_robodk()

    root = tk.Tk()
    root.title(“Suture Process")
    text_label = tk.Label(root, text="", wraplength=300)
    text_label.pack(padx=20, pady=20)
    suture_thread = threading.Thread(target=suture, args=(arduino, robot, gripper, needle
    suture_thread.daemon = True
    suture_thread.start()

    # Keyboard listener in main thread
    listener = keyboard.Listener(on_press=on_press, on_release=on_release)
    listener.start()

    root.mainloop()
    listener.stop()  # Ensure the listener stops when the Tkinter window is closed
    print("Suture process CLOSED!")
    print("Disconnecting Arduino...")
    arduino.close()

if __name__ == "__main__":
    main()
```



Suture process

Instruction

Moving the UR5e arm...

Gripper/Endowrist Orientation

R= 281.0   P= 0   W= 21

Gripper Torque

Torque Roll 1

Torque Roll 2

Torque Pitch

Torque Yaw

RoboDK python simulation program

```
def suture(arduino, robot, gripper, needle, base, text_label):

    while True:
            arduino.write(Command.GET_RPW.value)
            roll, pitch, yaw = [float(arduino.readline().strip()) for _ in range(3)]
            s1, s2 = [bool(int(arduino.readline().strip())) for _ in range(2)]
            torque_values = [float(arduino.readline().strip()) for _ in range(4)]
            if s1 and not s2:
                needle.setParentStatic(gripper)
                update_text_label(text_label, "Close Gripper")
            elif not s1 and not s2:
                gripper_pose = transl(Xg, Yg, Zg) * rotz(W) * roty(P) * rotx(R)
                gripper.setPose(gripper_pose)
                update_text_label(text_label, f"Mode 2. Gripper orientation: R={round(roll)} P={round(pitch)} W={round(ya
            elif not s1 and s2:
                needle.setParentStatic(base)
                update_text_label(text_label, "Open Gripper")
            elif key_states['u']:
                robot.MoveL(robot.Pose() * transl(0, 0, 10), False)
                key_states['u'] = False  # Reset the state after moving
                update_text_label(text_label, "Gripper moved up")
            elif key_states['d']:
                robot.MoveL(robot.Pose() * transl(0, 0, -10), False)
                key_states['d'] = False  # Reset the state after moving
                update_text_label(text_label, "Gripper moved down")
            elif key_states['e']:
                tcp_pose = transl(Xr,Yr,Zr) * rotz(ZERO_YAW) * rotz(W) * roty(P) * rotx(R)
                if robot.MoveL_Test(robot.Joints(), tcp_pose) == 0:
                    robot.MoveL(tcp_pose, True)
                    update_text_label(text_label, f"Mode 1. Robot orientation: R={round(roll)} P={round(pitch)} W={round(yaw)}")
                else:
                    update_text_label(text_label, "Mode 1. Robot orientation: Robot cannot reach the position")
                key_states['e'] = False  # Reset the state after moving
            else:
                update_text_label(text_label, f"Waiting: R={round(roll)} P={round(pitch)} W={round(yaw)}")
```

S2 Close

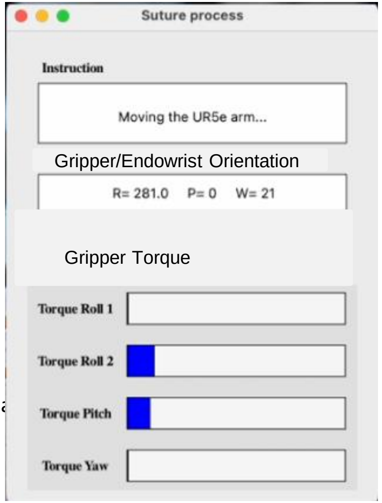
S1 S2 gripper mode

S1 Open

“u” backward

“d” forward

“e” Endowrist mode



Suture process

Instruction

Moving the UR5e arm...

Gripper/Endowrist Orientation

R= 281.0 P= 0 W= 21

Gripper Torque

Torque Roll 1

Torque Roll 2

Torque Pitch

Torque Yaw

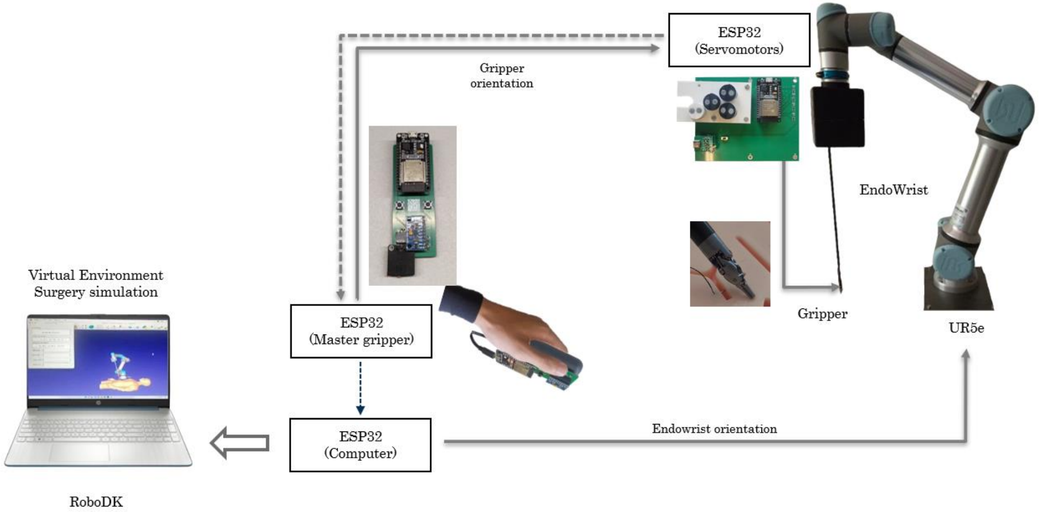
Lab session 1:

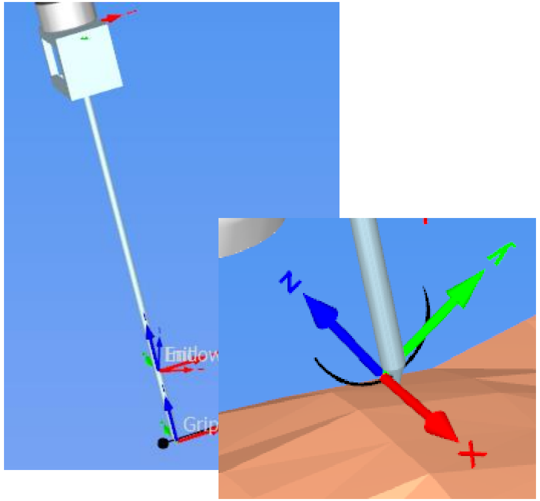
Previous Task:

Proposed roboDK suture process (**A3**)

Lab session:

Review system performances and practice the proposed suture process





Connect ESP32 computer to COM port
Connect ESP32 master to a battery
Connect ESP32 servos to USB-C Plug

Open roboDK Project
Open Lab folder Project in Vscode
Run the Project Python file

Practice the proposed simple suture process

Identify performances and limitations