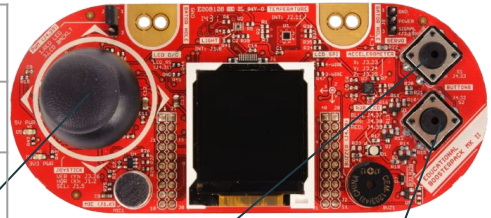
Explorer USS.0.1

== By Riccardo Ferro, Henrique Graça, ==
== Dennis Alberti, Tiago Silva ==

Link Explorer USS.0.1 video

ROVER & CAMERA - CONTROLLER

Action	Rover	Camera
UP ↑	Go Forward	Look Up
DOWN ↓	Go Backwards	Look Down
RIGHT →	Rotate Right	Look Right
LEFT ←	Rotate Left	Look Left
JOYSTICK BUTTON ⊗		Reset position to center



S2 Button - Change from TANK to CAMERA mode

S3 Button - Change the state machine

```
void _hwInit(){
    GPIO_setAsInputPinWithPullUpResistor(GPIO_PORT_P4, GPIO_PIN1);
    //Halting WDT and disabling master interrupts
    WDT_A_holdTimer();
    Interrupt_disableMaster();

    // Set the core voltage level to VCORE1
    PCM_setCoreVoltageLevel(PCM_VCORE1);

    // Set 2 flash wait states for Flash bank 0 and 1
    FlashCtl_setWaitState(FLASH_BANK0, 2);
    FlashCtl_setWaitState(FLASH_BANK1, 2);

    // Initializes Clock System
    CS_setDCOCenteredFrequency(CS_DCO_FREQUENCY_48);
    CS_initClockSignal(CS_MCLK, CS_DCOCLK_SELECT, CS_CLOCK_DIVIDER_1);
    CS_initClockSignal(CS_HSMCLK, CS_DCOCLK_SELECT, CS_CLOCK_DIVIDER_1);
    CS_initClockSignal(CS_SMCLK, CS_DCOCLK_SELECT, CS_CLOCK_DIVIDER_1);
    CS_initClockSignal(CS_ACLK, CS_REFOCLK_SELECT, CS_CLOCK_DIVIDER_1);
```

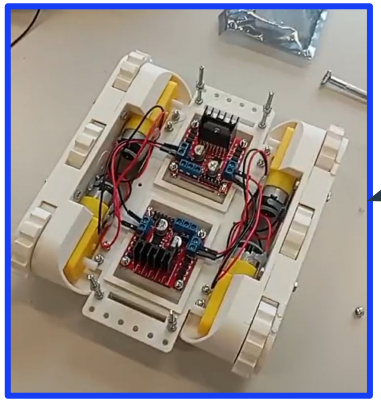
```
//tank movement
if (button2Toggle==0) {
    if ((resultsBuffer[1] < THRESHOLD_HIGH) && (resultsBuffer[1] > THRESHOLD_LOW+1000) &&
                                (resultsBuffer[0] < THRESHOLD_HIGH) && (resultsBuffer[0] > THRESHOLD
                            // If value y is greater than 12000 car m
                            // printf("Tank Forwards\n");
                    sendOnceTank(WriteDefault);}

    else if (resultsBuffer[1] > THRESHOLD_HIGH) {
    // If value y is greater than 12000 car moves forward
    //printf("Tank Forwards\n");
        Graphics_drawStringCentered(&g_sContext,(int8_t *)"  Forward ",AU
        sendOnceTank(WriteForward);
    } else if (resultsBuffer[1] < THRESHOLD_LOW) {
    // If value y is less than 1000 car moves backwards
    //printf("Tank Backwards\n");
        Graphics_drawStringCentered(&g_sContext,(int8_t *)" Backward ",AU
        sendOnceTank(WriteBackward);
    }
```

```
// **Button3 Toggle (Bottom button)**
int button3State = !(P3IN & GPIO_PIN5); // Read Bu

if (button3State && !button3PrevState) // Detect r
{
    mode = !mode; // Toggle state
    Graphics_clearDisplay(&g_sContext);
    printf("Mode %d:", mode);
}
button3PrevState = button3State; // Update previou
}
```

```
void _adcInit() {
    // Configure GPIO Pins for ADC Input P6.0 and P4.4 are set as peripheral input pins for the ADC module.
GPIO_setAsPeripheralModuleFunctionInputPin(GPIO_PORT_P6, GPIO_PIN0, GPIO_TERTIARY_MODULE_FUNCTION);
GPIO_setAsPeripheralModuleFunctionInputPin(GPIO_PORT_P4, GPIO_PIN4, GPIO_TERTIARY_MODULE_FUNCTION);

    // Enable and Initialize the ADC14 Module
ADC14_enableModule();
ADC14_initModule(ADC_CLOCKSOURCE_ADCOSC, ADC_PREDIVIDER_64, ADC_DIVIDER_8, 0);

    //Configure ADC for Multi-Sequence Mode allowing multiple channels to be read in sequence.
ADC14_configureMultiSequenceMode(ADC_MEM0, ADC_MEM1, true);

    //Configure ADC Conversion Memory ADC_MEM0 reads from analog input A15. ADC_MEM1 reads from analog input A9.
ADC14_configureConversionMemory(ADC_MEM0, ADC_VREFPOS_AVCC_VREFNEG_VSS, ADC_INPUT_A15, ADC_NONDIFFERENTIAL_INPUTS);
ADC14_configureConversionMemory(ADC_MEM1, ADC_VREFPOS_AVCC_VREFNEG_VSS, ADC_INPUT_A9, ADC_NONDIFFERENTIAL_INPUTS);

    //Enable ADC Interrupts
ADC14_enableInterrupt(ADC_INT1);
Interrupt_enableInterrupt(INT_ADC14);
Interrupt_enableMaster();

    //Start ADC Sampling and Conversion. Start continuous automatic conversions.
ADC14_enableSampleTimer(ADC_AUTOMATIC_ITERATION);
ADC14_enableConversion();
ADC14_toggleConversionTrigger(); // Start the ADC conversion
```

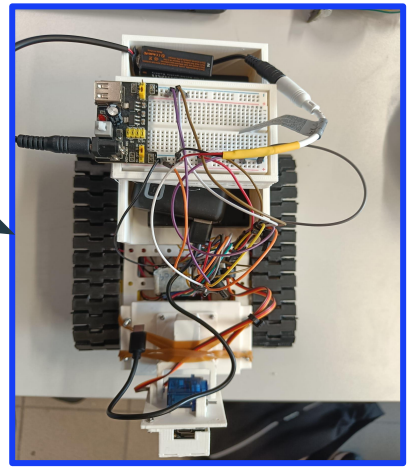
HARDWARE & STRUCTURE

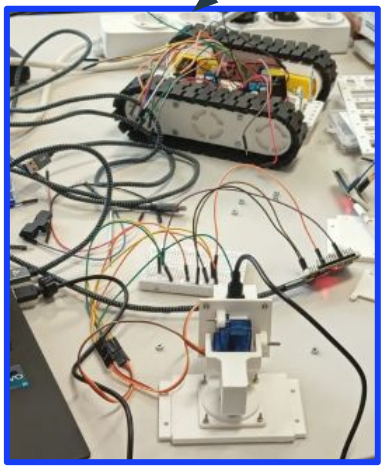
L298N drivers to control DC motors



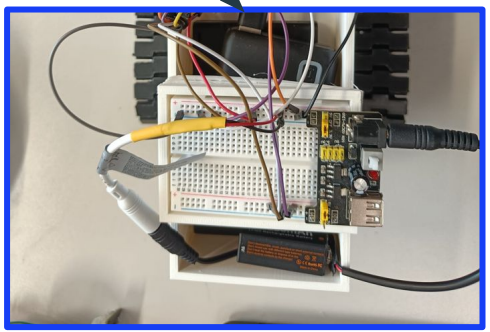
3D printed structure to ensure a light but resistant structure

4 different power modules, 3 for the rover and 1 for the Joystick





Servomotors directly connected to the ESP32



Voltage step down board to manage different voltage and power sources

MOTORS,DRIVERS AND CAMERA

- 2 output pin and 1 speed control pin;
- 2 rover modes: TANK (moving the dc motors) and CAMERA (moving the servomotors);
- Dc motor pins (motorXpinY) refers to driver pins;
- Servo motors pins are directly connected to ESP32;
- ESP32-Cam default template to have a working webserver.

```
void rightCam(){
  servoX.write(42);
}

void leftCam(){
  servoX.write(142);
}

void upCam(){
  servoY.write(115);
}

void downCam(){
  servoY.write (30);
}

void defaultCam(){
  servoX.write(87);
  servoY.write(63);
}
```

```
if (DATA == "left..") {
//Serial.println("COMMAND LEFT");
  leftstop();
  rightstop();
  rightforward();
  leftbackwards();
}
else if (DATA == "right.") {
    // Serial.println("COMMAND RIGHT");
  leftstop();
  rightstop();
  leftforward();
  rightbackwards();
}
```

```
else if (DATA == "cright") {
  rightCam();
}
else if (DATA == "cleft.") {
  leftCam();
}
else if (DATA == "down..") {
  downCam();
}
```

```
servoX.attach(13);
servoY.attach(32);
```

```
#define MOTOR1_SPEED 27
#define MOTOR2_SPEED 26
#define MOTOR3_SPEED 14
#define MOTOR4_SPEED 25
```

```
void leftforward(){
  digitalWrite(motor1pin1,HIGH);
  digitalWrite(motor1pin2,LOW);

  digitalWrite(motor2pin1,HIGH);
  digitalWrite(motor2pin2,LOW);
}

void rightforward(){
  digitalWrite(motor3pin1,LOW);
  digitalWrite(motor3pin2,HIGH);

  digitalWrite(motor4pin1,LOW);
  digitalWrite(motor4pin2,HIGH);
}

void leftbackwards(){
  digitalWrite(motor1pin1,LOW);
  digitalWrite(motor1pin2,HIGH);

  digitalWrite(motor2pin1,LOW);
  digitalWrite(motor2pin2,HIGH);
}

void rightbackwards(){
  digitalWrite(motor3pin1,HIGH);
  digitalWrite(motor3pin2,LOW);

  digitalWrite(motor4pin1,HIGH);
  digitalWrite(motor4pin2,LOW);
}
```

COMMUNICATION MSP

UART_A2 Module

~9600 baud rate

```
void UART_init(){
    GPIO_setAsPeripheralModuleFunctionInputPin(GPIO_PORT_P3,
                    GPIO_PIN2 | GPIO_PIN3, GPIO_PRIMARY_MODULE_FUNCTION);///Pin 3.2RX,pin3.3TX    || In ESP32 16RX 17TX
    CS_setDCOCenteredFrequency(CS_DCO_FREQUENCY_12);
    UART_initModule(EUSCI_A2_BASE, &uartConfig);
    UART_enableModule(EUSCI_A2_BASE);
}

void UART_SendString(const char *message) { //Needed because the UART send one character per time
    while (*message != '\0') { // Loop until null terminator is reached
        UART_transmitData(EUSCI_A2_BASE, *message); // Send the current character
        message++; // Move to the next character
    }
}
```

```
const eUSCI_UART_ConfigV1 uartConfig = {
        EUSCI_A_UART_CLOCKSOURCE_SMCLK,          // Clock Source: SMCLK  12MHz
        78,                                      // BRDIV = 78
        2,                                       // UCxBRF = 2
        0,                                       // UCxBRS = 0
        EUSCI_A_UART_NO_PARITY,                  // No parity
        EUSCI_A_UART_LSB_FIRST,                  // Least Significante Bit first
        EUSCI_A_UART_ONE_STOP_BIT,               // 1 stop bit
        EUSCI_A_UART_MODE,                       // UART mode
        EUSCI_A_UART_OVERSAMPLING_BAUDRATE_GENERATION // Oversampling. Defines de formula used to calculate de baud rate, using the first 4 parameter = 9554 (ap
};
```

```
void UART_init();
void UART_SendString(const char *message);
void resetCommandsTank();
void resetCommandsCamera();
void sendOnceTank(const char *message);
void sendOnceCamera(const char *message);
void sendAlternateMessage();
```

COMMUNICATION

```
#include <esp_now.h>
#include <WiFi.h>
```

```
typedef struct myData {
  String mensage;
} myData;
myData RecievedData;
```

```
uint8_t broadcastAddress[] = { 0xcc , 0xdb , 0xa7 , 0x34 , 0x29 , 0x14};
esp_now_peer_info_t peerInfo; //Saves the

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nLast Packet Send Status:\t");                                    //
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" : "Delivery Fail");  //
}
void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  Serial1.begin(9600, SERIAL_8N1, 16, 17); // opens a serial connection (BAUD RATE, RX, TX)
                                           //SERIAL_8N1 means that the serial communication w
                                           //This config should match the MSP on
  //Initialize ESP_now protocol
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  esp_now_register_send_cb(OnDataSent);
  memcpy(peerInfo.peer_addr, broadcastAddress, 6); //Saves and register the destination IP
  peerInfo.channel = 0;
  peerInfo.encrypt = false;
//Add the peer info
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer");
    return;
  }
}
```

ESP1

```
void loop() {
  //Loop to read 6 chars and tranform them into a string
  while (Serial1.available()) { //If there is something to read in Serial1
    recievedChar = Serial1.read();  // read a char from Serial1
    recievedString[bufferIndex] = recievedChar; //add to the String array
    bufferIndex++;
    // Verify if we are at the maximum length
    if (bufferIndex >= BUFFER_SIZE - 1) {
      recievedString[bufferIndex] = '\0';  // Adds the null char to finish the String
      bufferIndex = 0;
    }
  }
  mydata.mensage = recievedString; //Saves the recieved string into the sending structure
  //Send the data and saves the output in "result"
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &mydata, sizeof(mydata));
```

ESP2

```
WiFi.mode(WIFI_STA);
//Here we initialize the ESP-NOW protocol
if (esp_now_init() != ESP_OK) {
  //Serial.println("Error initializing ESP-NOW");
  return;
}
// Once ESPNow is successfully Init, we will register for recv CB to
// get recv packer info
esp_now_register_recv_cb(esp_now_recv_cb_t(OnDataRecv));
```

```
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  memcpy(&RecievedData, incomingData, sizeof(RecievedData));
  //Serial.print("Mensage: ");
  //Serial.println(myData.mensage);

  DATA = RecievedData.mensage;
```

TESTING & FUTURE IMPROVEMENT

The code was highly hardware dependent, so the tests has been done step by step, checking the results of little pieces of code, compiled and run on the respective IDE (Code Composer, Arduino IDE). Then, when the code was correct and ready, it was implemented in the final one to test them all together.

- Implementing some led light, to use the rover also in darker places;
- Implementing a more detailed interface on screen, by creating a menu screen to navigate through the different modes;
- Implementing a timer to register inactivity from the user, causing the system to switch to a sleep mode;
- Using the Pulse Width Modulation to regulate the movement speed based on the position of the controller (so constantly changing , and not a fixed value);
- Optimize the code; make the code easier to read, more intuitive.

Testing example

```
} else if (resultsBuffer[1] < THRESHOLD_LOW) {
    // If value y is less than 1000 car moves backwards
    //printf("Tank Backwards\n");
    Graphics_drawStringCentered(&g_sContext,
                                (int8_t *)" Backward ",
                                AUTO_STRING_LENGTH,
                                64,
                                40,
                                OPAQUE_TEXT);
}
```