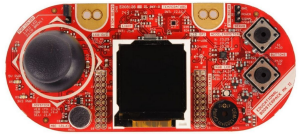
Embedded Software of the Internet of Things - Group Project

# Claw Machine

By Luca Podavini, Alberto Cimmino, Angela Hu and Sara Tait

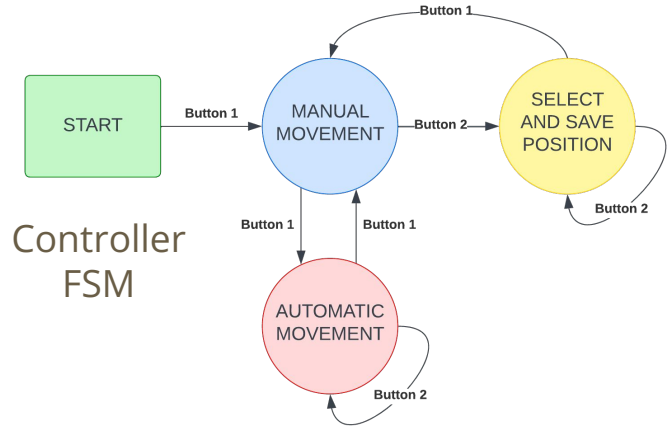
Link to YouTube demo

# Controller



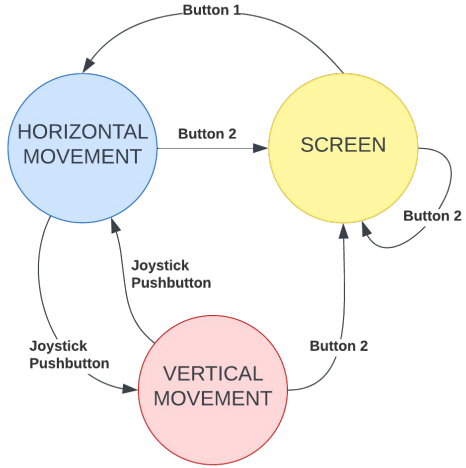
CONTROLLER → BLUETOOTH MASTER→ BLUETOOTH SLAVE → MOTORS

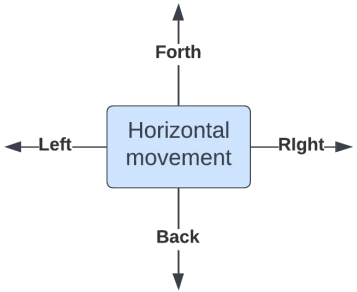
→ SCREEN

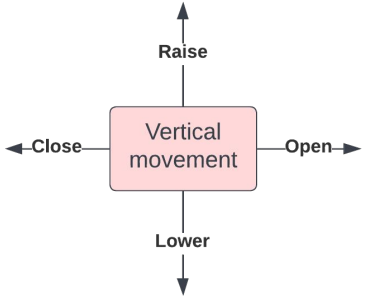



Controller FSM

Joystick movement modes



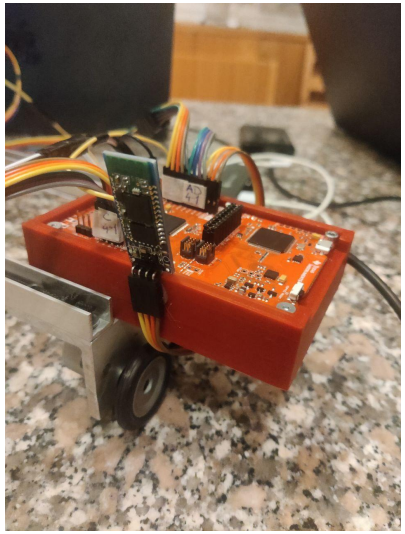



# Hardware + Structure

Foldable and portable aluminium frame

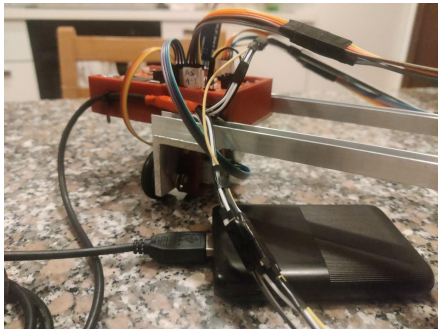
28BYJ-48 stepper motors with ULN2003 driver board

3D printed supports

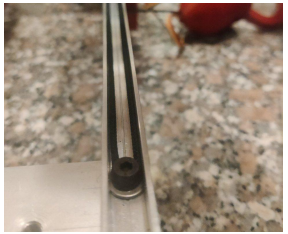


Gripper with servo

Separate circuit for USB power supply (from the powerbank to the black and white wires)



Inner lining of the guide with a rope for smoothness



# Driver software & motors control

```
// function that store the four steps to properly spin the motor.
Step_t step(StepNumber_t stepNumber) {
    switch(stepNumber) {                                    // Step COIL_1  COIL_2  COIL_3  COIL_4
        case 0:
            return (Step_t) {HIGH, HIGH, LOW, LOW};         // 0    HIGH    LOW     HIGH    LOW
        case 1:
            return (Step_t) {LOW, HIGH, HIGH, LOW};         // 1    LOW     HIGH    HIGH    LOW
        case 2:
            return (Step_t) {LOW, LOW, HIGH, HIGH};         // 2    LOW     HIGH    LOW     HIGH
        case 3:
            return (Step_t) {HIGH, LOW, LOW, HIGH};         // 3    HIGH    LOW     LOW     HIGH
        default:
            return (Step_t) {};
    }
}
```

- 4 output pins
- output voltage with a specific pattern
- output updated using a timer
- the delay between two steps control the speed (from 2.5ms to 5ms)
- two movement modes: manual and target

```
const int MIN_SPEED_DELAY = 20; // the delay to spin a stepper at its minimum speed 20 extra tick to wait, 5ms period
const int MAX_SPEED_DELAY = 10; // the delay to spin a stepper at its maximum speed. 10 extra tick to wait, 2.5ms period

// timer configuration for the stepper main loop
const Timer_A_UpModeConfig timerConfig = {
        .clockSource = TIMER_A_CLOCKSOURCE_SMCLK,                                   // SMCLK Clock Source
        .clockSourceDivider = TIMER_A_CLOCKSOURCE_DIVIDER_4,                        // SMCLK/4 = 24MHz/4 = 6MHz
        .timerPeriod = 1500,                                                        // Timer period: 0.25ms
        .timerInterruptEnable_TAIE=TIMER_A_TAIE_INTERRUPT_DISABLE,                  // Disable Timer interrupt
        .captureCompareInterruptEnable_CCR0_CCIE= TIMER_A_CCIE_CCR0_INTERRUPT_ENABLE,   // Enable CCR0 interrupt
        .timerClear=TIMER_A_DO_CLEAR                                                // Clear value
};
```

```
// motors control loop
void TA1_0_IRQHandler(void)
{
    // clear interrupt flag
    Timer_A_clearCaptureCompareInterrupt(TIMER_A1_BASE, TIMER_A_CAPTURECOMPARE_REGISTER_0);
    // move claw machine
    Claw_tryMove(&clawMachine);
}
```

```
// move the claw machine according to its status and its motor status
void Claw_tryMove(ClawMachine_t *clawMachine)
{
    if (clawMachine->mode.tag == TARGET_MODE)
    {
        // if in target mode move the two carts to their respective target
        int res1 = Cart_goTo(&clawMachine->cart_b, clawMachine->mode.data.target_mode.target.x);
        int res2 = Cart_goTo(&clawMachine->cart_a, clawMachine->mode.data.target_mode.target.y);
    }
    else
    {
        // if in manual mode move the two carts, the whinch and the gripper if they has to
        Cart_tryMove(&clawMachine->cart_a);
        Cart_tryMove(&clawMachine->cart_b);

        Stepper_tryMove(&clawMachine->whinch);
        Servo_tryMove(&clawMachine->gripper);
    }
}
```

Servo motor controlled using PWM:

- signal period of 20ms
- duty cycle from 1ms to 2ms

```
// timer settings for servo PWM
Timer_A_UpModeConfig servoTimerPWMConfig = {
        TIMER_A_CLOCKSOURCE_SMCLK,              // SMCLK = 24 MHz
        TIMER_A_CLOCKSOURCE_DIVIDER_12,         // SMCLK/12 = 2MHz
        40000,                                  // 20 ms tick period
        TIMER_A_TAIE_INTERRUPT_DISABLE,         // Disable Timer interrupt
        TIMER_A_CCIE_CCR0_INTERRUPT_DISABLE,    // Disable CCR0 interrupt
        TIMER_A_DO_CLEAR                        // Clear value
};
```

# Communication (Bluetooth) and command encoding

Controller Side:

- hc05 master
- encodes commands into a byte
- operations at bit level
- send commands over bluetooth

```
uint8_t SetNbitToOne(uint8_t value, int n)
{
    return (value | (1 << n));
}

uint8_t encodeVelocity(uint8_t velocity, uint8_t value)
{
    velocity = velocity << 4;

    return (value | velocity);
}

uint8_t encodeTarget(uint8_t target, uint8_t value)
{
    target = target << 2;
    return (value | target);
}
```

Claw Machine Side

- hc06 slave
- when data is received, reads and decodes the commands
- sends commands to motors

```
//UART handler for EUSCIA2
void EUSCIA2_IRQHandler(void)
{
    uint8_t c;
    uint32_t status = MAP_UART_getEnabledInterruptStatus(EUSCI_A2_BASE);
    MAP_UART_clearInterruptFlag(EUSCI_A2_BASE, status);

    if(status & EUSCI_A_UART_RECEIVE_INTERRUPT)
    {
        c = MAP_UART_receiveData(EUSCI_A2_BASE);

        interpretCommand(c, &clawMachine);
        Interrupt_disableSleepOnIsrExit(); //
    }
}
```

# Testing and future improvements

- **Hardware Independent testing**
  - Encoding and decoding of Bluetooth message
  -
- **Hardware Dependent testing**
  - Used Debug view for motors
  - #if DEBUG

- **Future improvements**
  - Better bluetooth communication
  - Stronger motors
  - Better shaped gripper

```
# if DEBUG
uint8_t data[1];
while (1)
{
    ReadFromBlueTooth(data);

    InterpretCommand(data[0]);
}
#endif
```