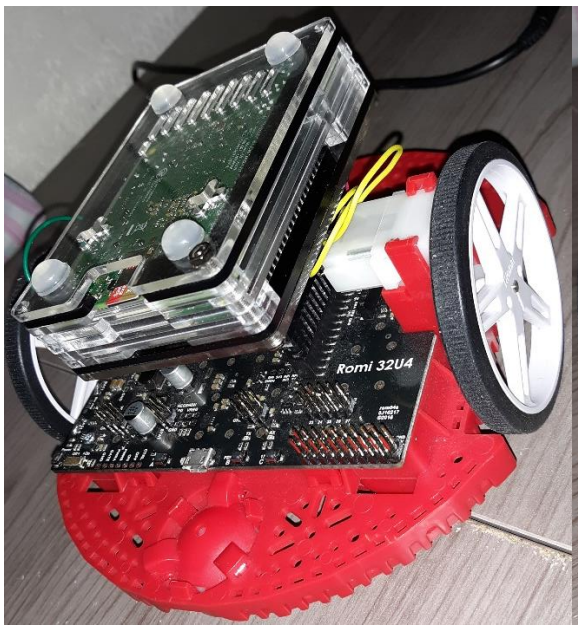
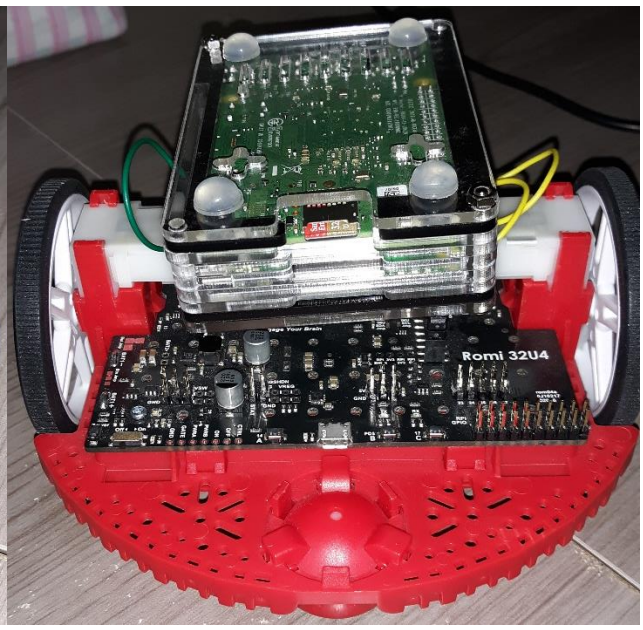
Assignment 3

```
#include <Servo.h>
#include <Romi32U4.h>
#include <PololuRPiSlave.h>

/* This example program shows how to make the Romi 32U4 Control Board
 * into a Raspberry Pi I2C slave.  The RPi and Romi 32U4 Control Board can
 * exchange data bidirectionally, allowing each device to do what it
 * does best: high-level programming can be handled in a language such
 * as Python on the RPi, while the Romi 32U4 Control Board takes charge
 * of motor control, analog inputs, and other low-level I/O.
 *
 * The example and libraries are available for download at:
 *
 * https://github.com/pololu/pololu-rpi-slave-arduino-library
 *
 * You will need the corresponding Raspberry Pi code, which is
 * available in that repository under the pi/ subfolder.  The Pi code
 * sets up a simple Python-based web application as a control panel
 * for your Raspberry Pi robot.
 */

// Custom data structure that we will use for interpreting the buffer.
// We recommend keeping this under 64 bytes total.  If you change the
// data format, make sure to update the corresponding code in
// a_star.py on the Raspberry Pi.

struct Data
{
  bool yellow, green, red;
  bool buttonA, buttonB, buttonC;

  int16_t leftMotor, rightMotor;
  uint16_t batteryMillivolts;
  uint16_t analog[6];

  bool playNotes;
  char notes[14];

  int16_t leftEncoder, rightEncoder;
};

PololuRPiSlave<struct Data,5> slave;
PololuBuzzer buzzer;
Romi32U4Motors motors;
Romi32U4ButtonA buttonA;
Romi32U4ButtonB buttonB;
Romi32U4ButtonC buttonC;
```

```
Romi32U4Encoders encoders;
void setup()
{
  // Set up the slave at I2C address 20.
  slave.init(20);

  // Play startup sound.
  buzzer.play("v10>>g16>>>c16");
}

void loop()
{
  // Call updateBuffer() before using the buffer, to get the latest
  // data including recent master writes.
  slave.updateBuffer();

  // Write various values into the data structure.
  slave.buffer.buttonA = buttonA.isPressed();
  slave.buffer.buttonB = buttonB.isPressed();
  slave.buffer.buttonC = buttonC.isPressed();

  // Change this to readBatteryMillivoltsLV() for the LV model.
  slave.buffer.batteryMillivolts = readBatteryMillivolts();

  for(uint8_t i=0; i<6; i++)
  {
    slave.buffer.analog[i] = analogRead(i);
  }

  // READING the buffer is allowed before or after finalizeWrites().
  ledYellow(slave.buffer.yellow);
  ledGreen(slave.buffer.green);
  ledRed(slave.buffer.red);
  motors.setSpeeds(slave.buffer.leftMotor, slave.buffer.rightMotor);

  // Playing music involves both reading and writing, since we only
  // want to do it once.
  static bool startedPlaying = false;

  if(slave.buffer.playNotes && !startedPlaying)
  {
    buzzer.play(slave.buffer.notes);
    startedPlaying = true;
  }
  else if (startedPlaying && !buzzer.isPlaying())
  {
    slave.buffer.playNotes = false;
    startedPlaying = false;
```

```
  }
  slave.buffer.leftEncoder = encoders.getCountsLeft();
  slave.buffer.rightEncoder = encoders.getCountsRight();

  // When you are done WRITING, call finalizeWrites() to make modified
  // data available to I2C master.
  slave.finalizeWrites();
}
```





Video link:
[Assignment 3 - YouTube]

GitHub link:
[Mobile-Robotics/Assignment3 at master · MeralAbuJaser/Mobile-Robotics (github.com)]