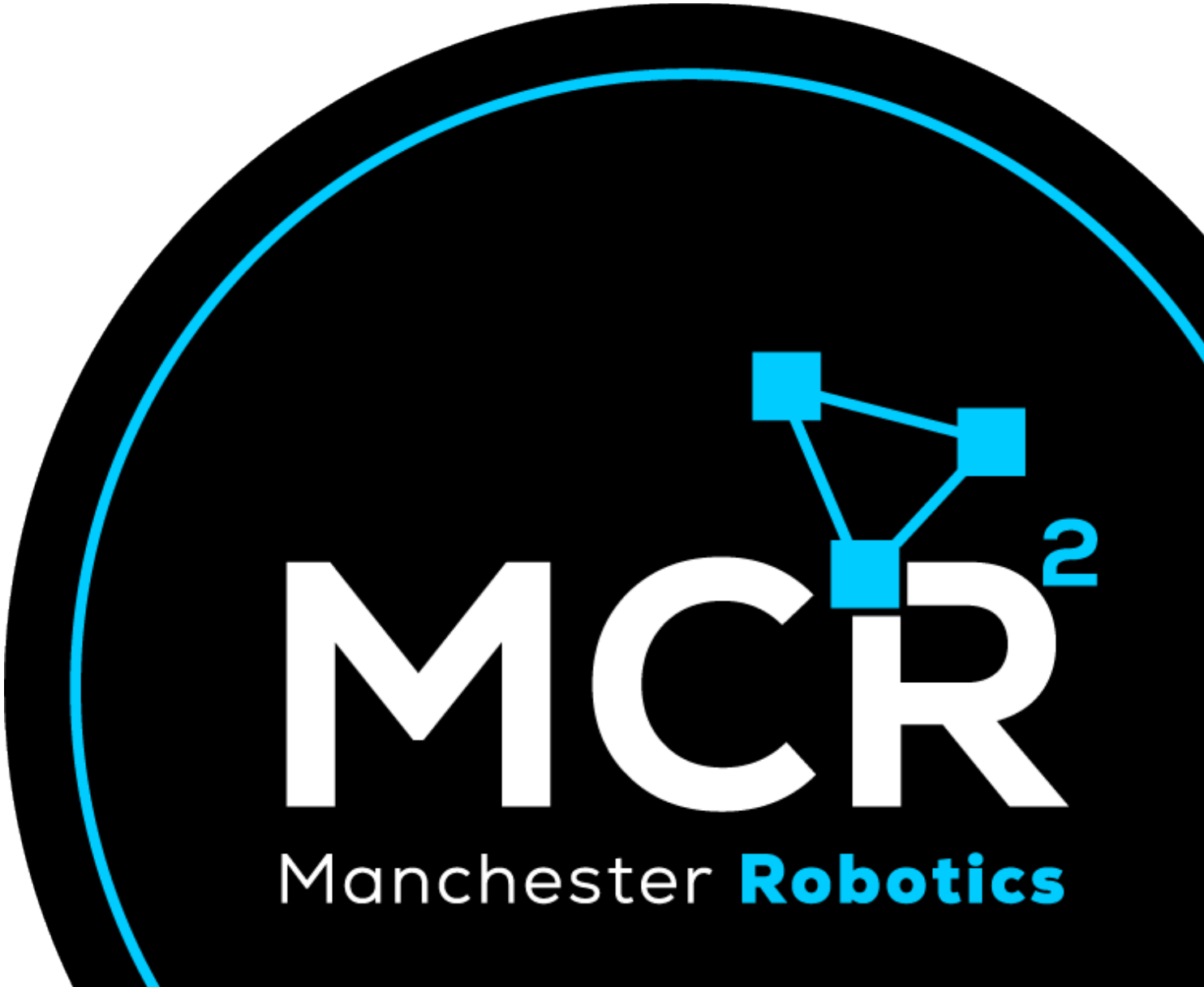
Challenges

Mini challenge 3

{Learn, Create, Innovate};

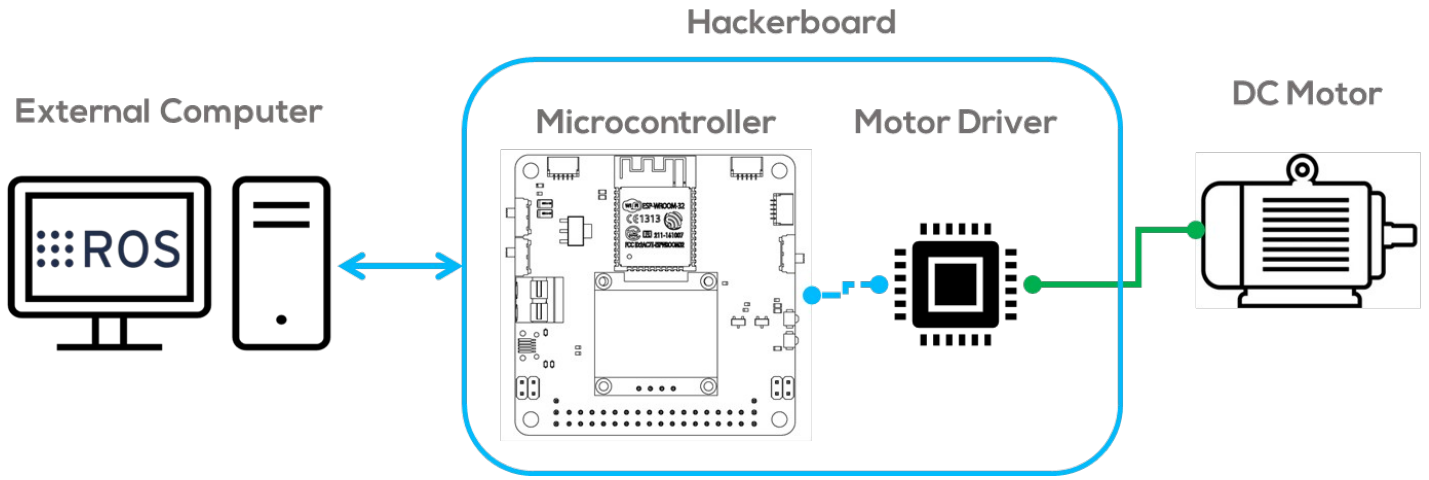


Mini Challenge 3

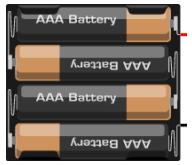
Introduction

This mini challenge is intended for the student to review the concepts introduced in the previous sessions.

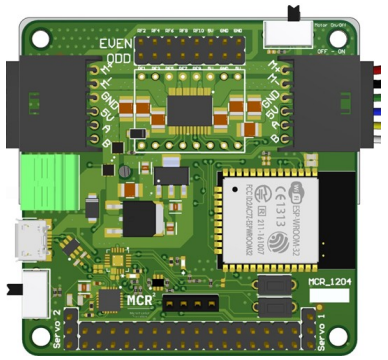
- The activity consists of creating several ROS nodes to regulate the speed of a DC Motor.
- The motor will be controlled using an external computer, a microcontroller, and a motor driver.
    - See following slide for requirements.



Mini Challenge 3: Requirements

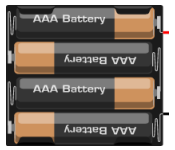


Battery Pack
5 - 12 V

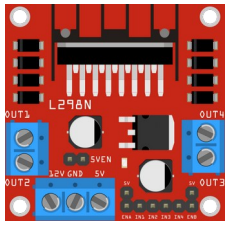


Hackerboard

6 VDC
Brushed
Motor



Battery Pack
5 - 12 V



Motor Driver
L298n

Wires
(Dupont or
any wire)

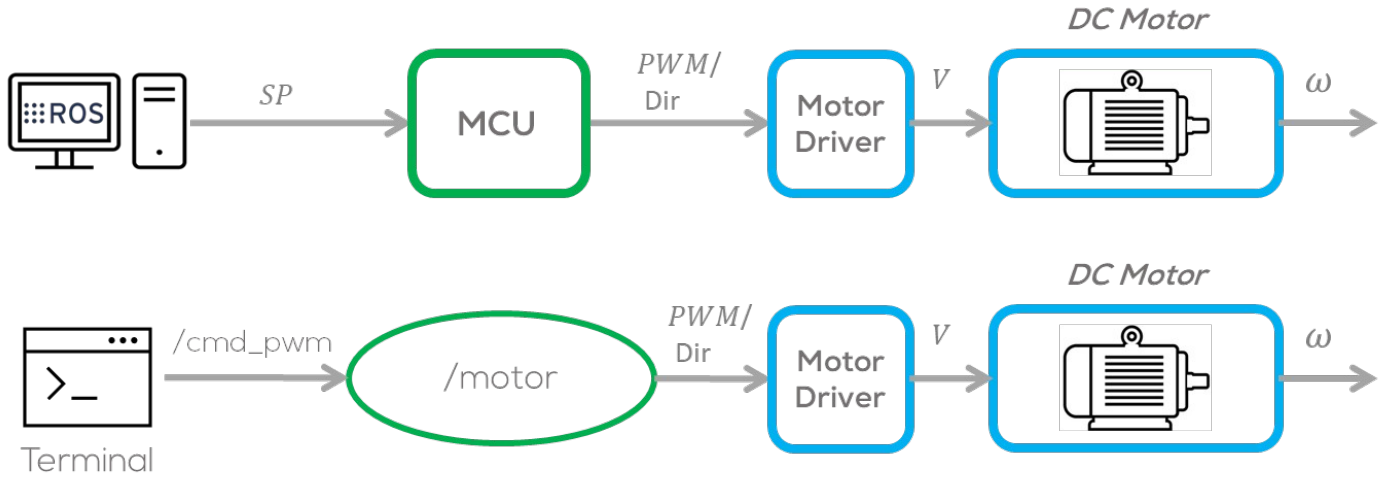
ESP32

6 VDC
Brushed
Motor

Mini Challenge 3

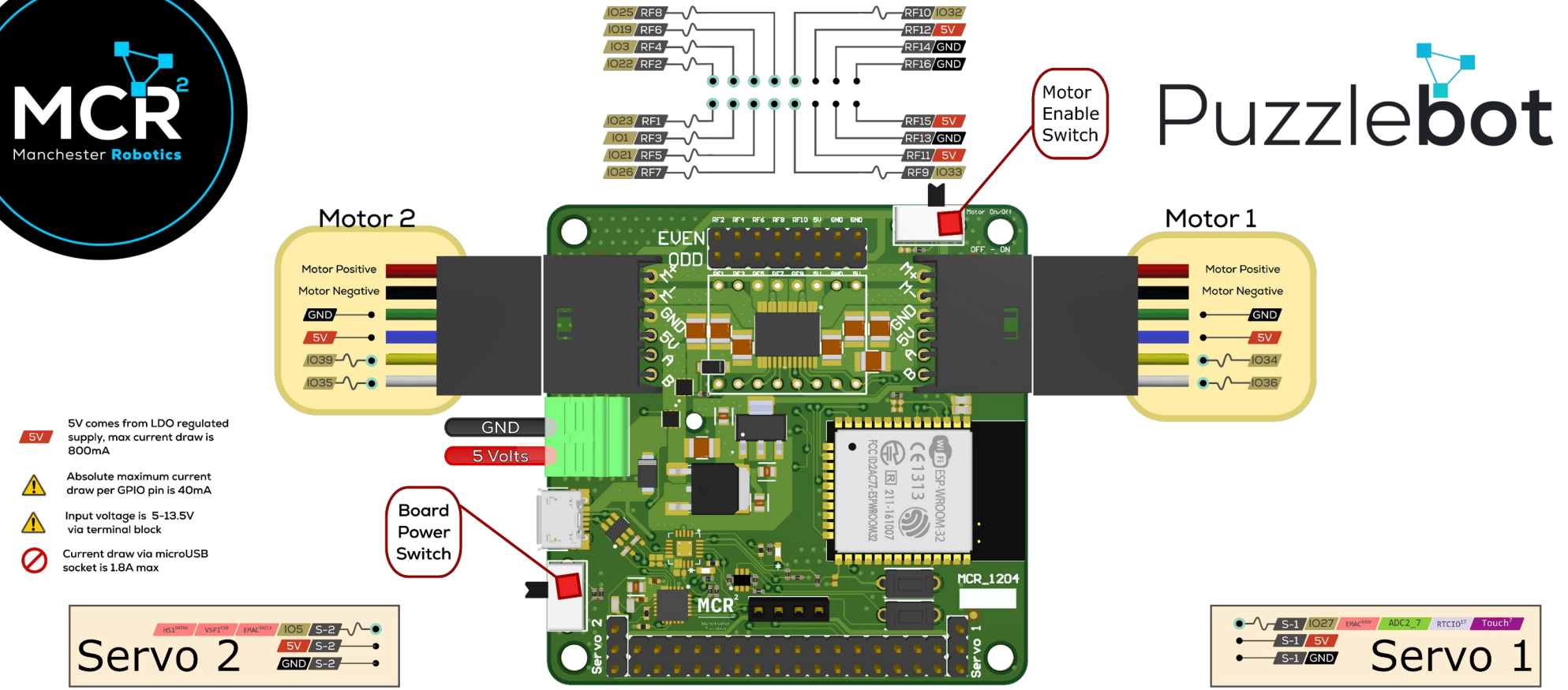
Motor Node

- The “/motor” node must run on the MCU (Hackerboard or ESP32)
- Must subscribe to the topic “/cmd_pwm”.
- The output of the ESP32 must be a “PWM signal” and a direction signal to the Motor Driver.
  - The PWM duty cycle and direction must be mapped to the interval , where the sign represents the direction.
  - Hackerboard already include the motor driver.
- The Motor driver must output the required power to the DC Motor.
  - See next slide for connection diagrams.



MCR² Manchester Robotics

Puzzlebot

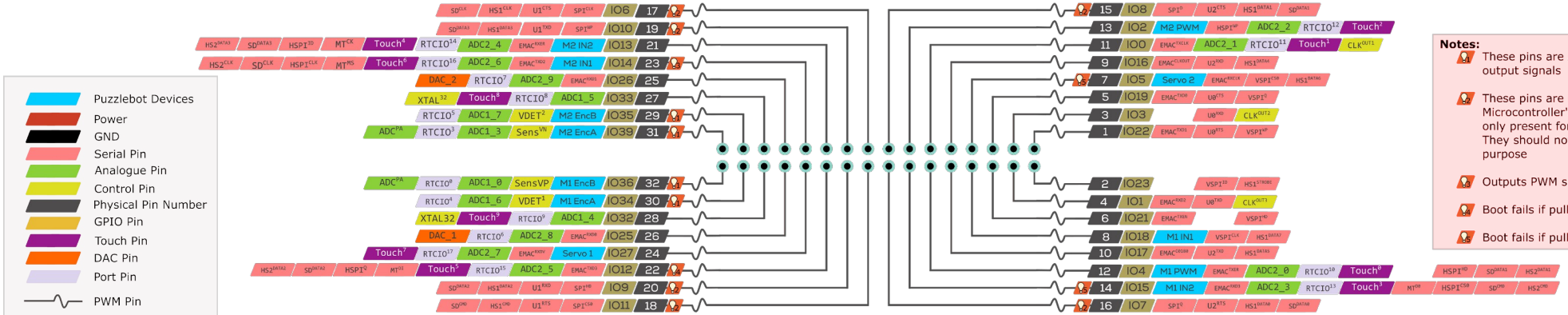


5V comes from LDO regulated supply, max current draw is 800mA

Absolute maximum current draw per GPIO pin is 40mA

Input voltage is 5-13.5V via terminal block

Current draw via microUSB socket is 1.8A max



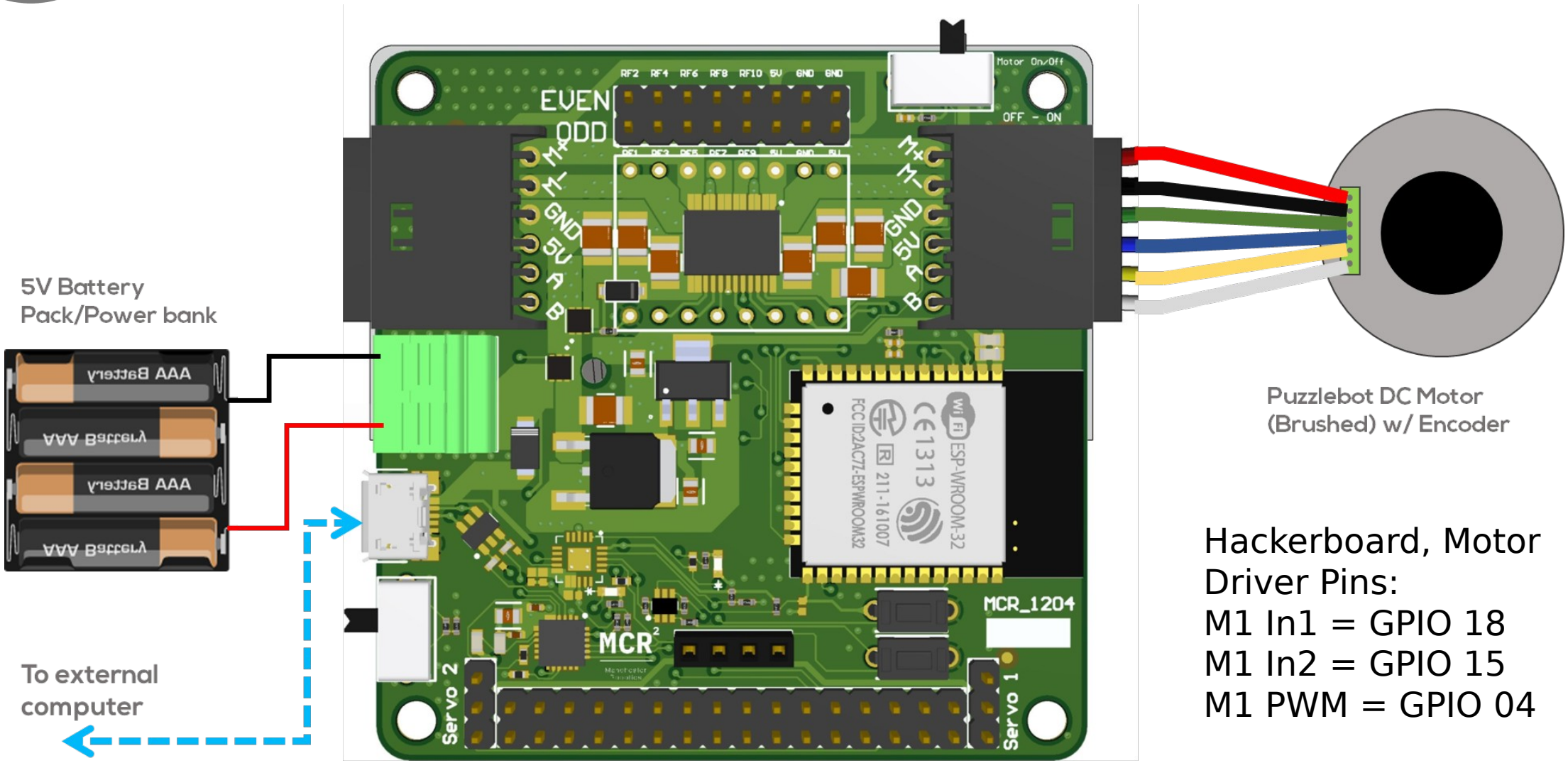
Notes:

- These pins are ***input only***. They cannot output signals
- These pins are integrated to the Microcontroller's flash memory, and are only present for debugging purposes. They should not be used for general purpose
- Outputs PWM signal at boot time
- Boot fails if pulled high
- Boot fails if pulled low

https://www.manchester-robotics.com

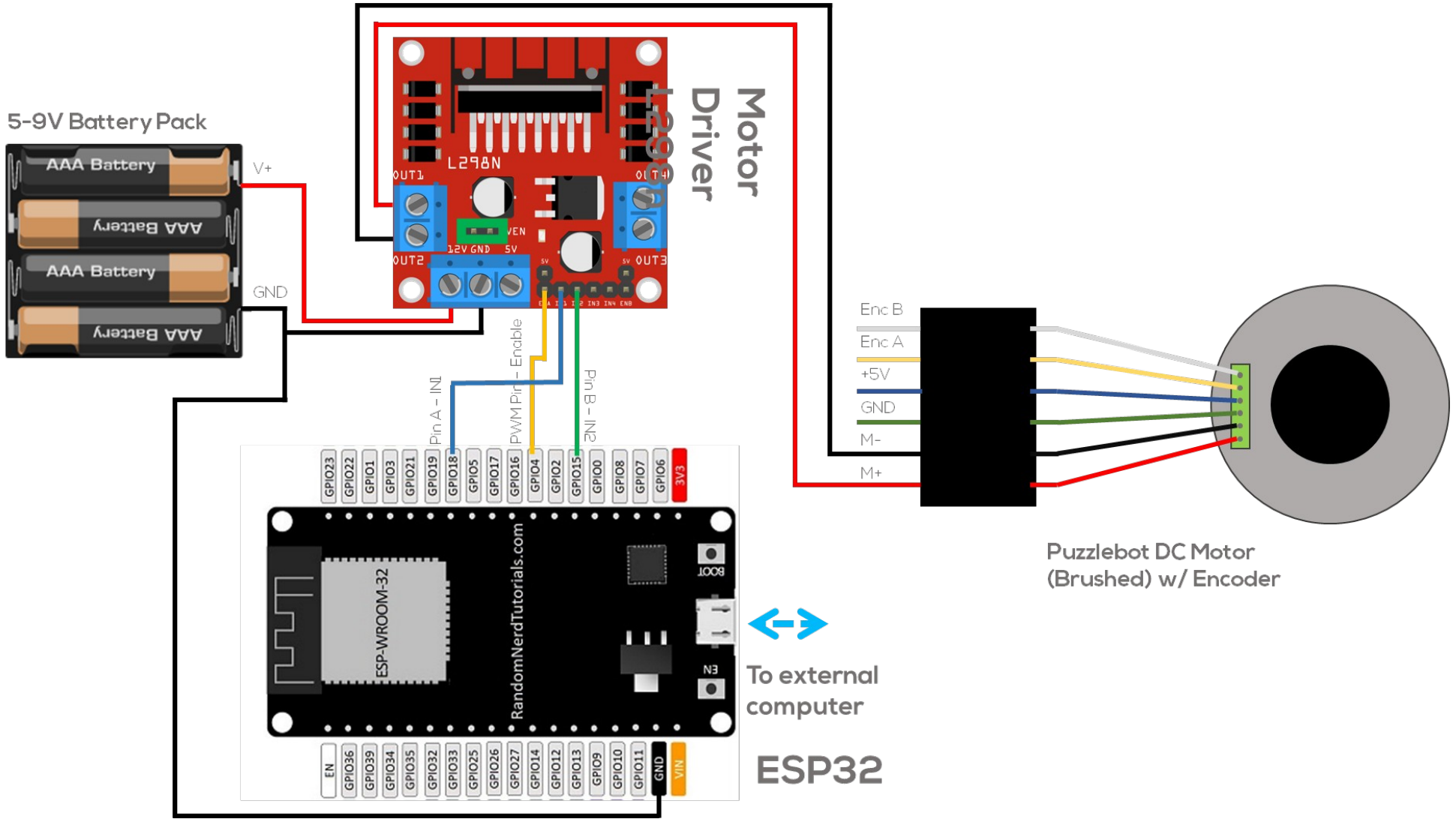
01 FEB 2022
v1

Mini Challenge 3: Connection Diagrams



Hackerboard, Motor Driver Pins:
M1 In1 = GPIO 18
M1 In2 = GPIO 15
M1 PWM = GPIO 04

Mini Challenge 3: Connection Diagrams



Mini Challenge 3

Hints (ESP32/Hackerboard PWM)

- The ESP 32 does not use the “analogwrite()” command to generate a PWM signal.
- To generate a PWM signal the following steps must be followed.

1. Choose a PWM channel. There are 16 channels from 0 to 15.
2. Set the PWM signal frequency (980 Hz standard).
3. Set the signal’s duty cycle resolution. From 1 to 16 bits resolution (8 bits typical, 0 to 255 resolution).
4. Specify the output GPIO.
5. More information can be found here.

```
void setup(){

ledcSetup(ledChannel, freq, resolution);

ledcAttachPin(ledPin, ledChannel);

}


void loop(){

ledcWrite(ledChannel, dutyCycle);

delay(15);

 }
```

Rules

- This is challenge **not** a class. The students are encouraged to research, improve tune explain their algorithms by themselves.
- MCR2(Manchester Robotics) Reserves the right to answer a question if it is determined that the questions contains partially or totally an answer.
- The students are welcomed to ask only about the theoretical aspect of the classed.
- No remote control or any other form of human interaction with the simulator or ROS is allowed (except at the start when launching the files).
- It is **forbidden** to use any other internet libraires with the exception of standard libraires or NumPy.
- If in doubt about libraires please ask any teaching assistant.
- Improvements to the algorithms are encouraged and may be used as long as the students provide the reasons and a detailed explanation on the improvements.
- All the students must be respectful towards each other and abide by the previously defined rules.
- Manchester Robotics reserves the right to provide any form of grading. Grading and grading methodology are done by the professor in charge of the unit.