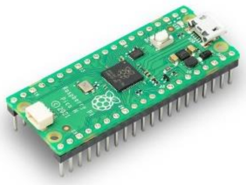
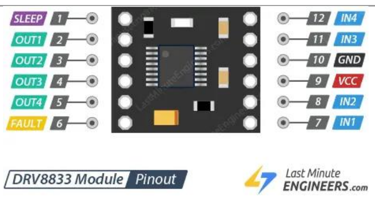
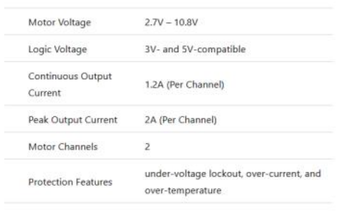
HERIOT-WATT
ROBOTICS SOCIETY

Robotics Society – Kart Workshop

Session 2 – Electronics

Last week, you began assembly of the karts, putting together the laser cut frame and beginning the wheel assemblies depending on your design. Today we will be completing the initial electronics setup, wiring up the Raspberry Pi Pico, DRV8833 motor controller, and depending on your choice of kart, the wheels or servo. We don’t include a direct wiring diagram in here – that’d be too easy! But you can compare against the working kart in the sessions.

Part list

Part name	Description/use	Picture
Raspberry Pi Pico W **Documentation:**	The main controller for the system, receiving input from its onboard Bluetooth chip, processing the input signals, and outputting appropriate PWM signals to the motor controller. The Pico itself can handle an **input voltage of up to around 5.5V,** though we recommend keeping it slightly below this. The whole board can be programmed in multiple languages, for our workshop we will be using MicroPython. Though it can be programmed directly in C, the software that interfaces with the Bluetooth capabilities is much more complex in C than Micropython.	
DRV8833 Motor Controller **Documentation:**	This board receives the PWM signals from the Pico and controls the motor forward and backwards at varying speeds. Using a motor controller is important, as the Pico is not designed to control motors nor servos directly off its internal voltage regulator. Motors and servos can draw large currents quickly, for example while changing directions. They can also produce back EMF, which the Pico will not be expecting or able to handle without additional components. The motor controller solves these problems and combines the solution into a single board	 SLEEP 1 OUT1 2 OUT2 3 OUT3 4 OUT4 5 FAULT 6 12 IN4 11 IN3 10 GND 9 VCC 8 IN2 7 IN1 DRV8833 Module Pinout Last Minute ENGINEERS.com 

Motor Voltage	2.7V – 10.8V
Logic Voltage	3V- and 5V-compatible
Continuous Output Current	1.2A (Per Channel)
Peak Output Current	2A (Per Channel)
Motor Channels	2
Protection Features	under-voltage lockout, over-current, and over-temperature

Wiring Description

To control the motors effectively, we need to supply the motor controller with 4 possible PWM signals. PWM is effectively turning a signal on and off very fast, and with this switching, the time that the device is on will stabilise to an overall lower voltage. For more information and answers to “why cant we just directly turn the voltage down”, see here:

In our sample program, we output this from GP0-3. It should be noted, that when using PWM signals from the Pico, care must be taken to avoid trying to set different frequencies on the same channel. The Pico has **8 independent PWM generators** called slices. Each slice has two channels (A and B), which makes a total of 16 PWM channels. While different outputs can be sent to each channel, the frequency is set per slice. This is only important for us, as the PWM frequency for the motors versus the servo will be different. More information is available in the Pico documentation, linked at the top of this document.

Our motor controller makes use of the dual h-bridge driver, the DRV8833, and can handle up to a maximum of 10.8V. The DRV8833 also contains several protection features, such as under-voltage lockout, over-current, and over-temperature protection. Each of these events will disable the device, and will only resume its operation after the fault has been removed. To learn more about the H-bridge drivers and motor controller work internally, see the documentation linked in the table above.

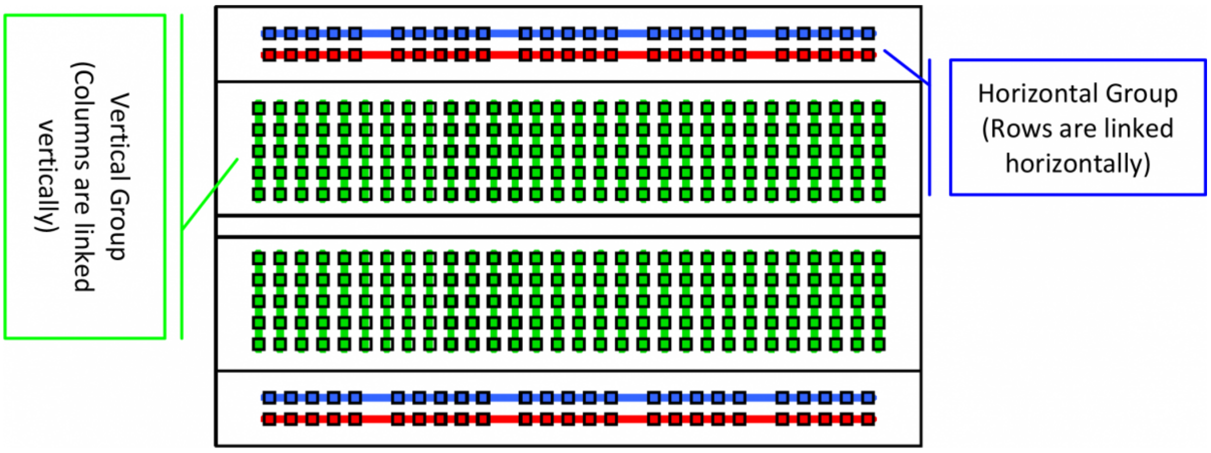
The board itself is made up of 12 pins, with 4 dedicated to input PWM signals, 4 dedicated to output motor control, and 2 for power input (Note: This motor controller uses the same motor and logic voltage, thus only one power supply for the whole unit is required). Additional pins are not used in our use case, but are described in the docs above.

Four input pins on the controller receive the 4 PWM signals that we generate above from the Pico. These control both the speed and direction of the motor. A table below shows how changing the levels of these input pins controls the motor: (Note: One motor is controlled using IN1 & IN2, and the other motor controlled by IN3 & IN4).

IN1/IN3	IN2/IN4	Spinning Direction
Low(0)	Low(0)	Motor OFF
High(1)	Low(0)	Forward
Low(0)	High(1)	Reverse
High(1)	High(1)	Motor OFF

Step 1

Press the Pico into the breadboard, with pins 1 and 40 touching one end of the board. Verify this by ensuring the MicroUSB connector is at the edge of one of the breadboards. Breadboards are connected in lines both horizontally and vertically. See the diagram below for examples of which pins are grouped together. Remember, you want each pin that the Pico presses into the breadboard to be on its own group (a green group), no two pins should be grouped together. The red and blue groups at either side of the breadboard are called “rails”, and they are used for power, with red corresponding to your positive voltage (~5V for us), and blue corresponding to 0V (or ground).


Vertical Group (Columns are linked vertically)
Horizontal Group (Rows are linked horizontally)

After the Pico is correctly placed into the breadboard, press the DRV8833 board into the available end of the breadboard. First, check the bottom of the motor controller, and see which side is labelled with the pins GND and VCC. Place these pins facing the same side as pin 40 of the Pico (remember, the pins on the DRV are on the bottom of the board, so try not to flip it around when placing it in!)

Connect the GP0-3 pins from the Pico in order to the IN1-4 pins on the motor controller, using jumper wires. If the wires are too long, don’t worry, we can find shorter wires later to improve the aesthetics. Verify each jumper connects as follows:

- GP0 – IN1
- GP1 – IN2
- GP2 – IN3
- GP3 – IN4

Step 2

Connect the required 5v and ground pins to both the motor controller and Pico. These should come from your 5V and 0V rails running alongside your breadboard (however don’t worry this rail isn’t powered until we plug in our battery next week).

With the Pico, connect a jumper wire from pin 39 (VSYS) to the red 5V rail on the side of your breadboard; then from pin 38 (GND) to the blue rail on the side of the breadboard. Ensure you plug into VSYS, as this pin accepts an external system voltage in between 1.8V and 5.5V

Similarly, connect pins VCC on the motor controller to the red 5V rail, and the GND pin to the 0V blue rail on the side of the breadboard.

Step 3

Connect the motor wires that you crimped previously to the motor controller. These pins are the center 4 on the side of the motor controller that you haven't used yet. Ensure that one motor is connected to pins OUT1 & OUT2, then the other motor to OUT3 & OUT4. Which pins are plugged into which OUT between 1&2 doesn't matter yet, we will likely have to adjust it later.

Step 4 (racer karts only with servo)

Jumper wires or recrimping the servo wires to male ends can be chosen to connect the servo lead end to the breadboard. We would recommend just using long jumper wires for now, and then recrimping in time.

The servo has three input wires: red for positive voltage, black for ground (0V), and orange for signal.

Plug the red and black wires directly into the red and blue rails on your breadboard, corresponding red to red and blue to black.

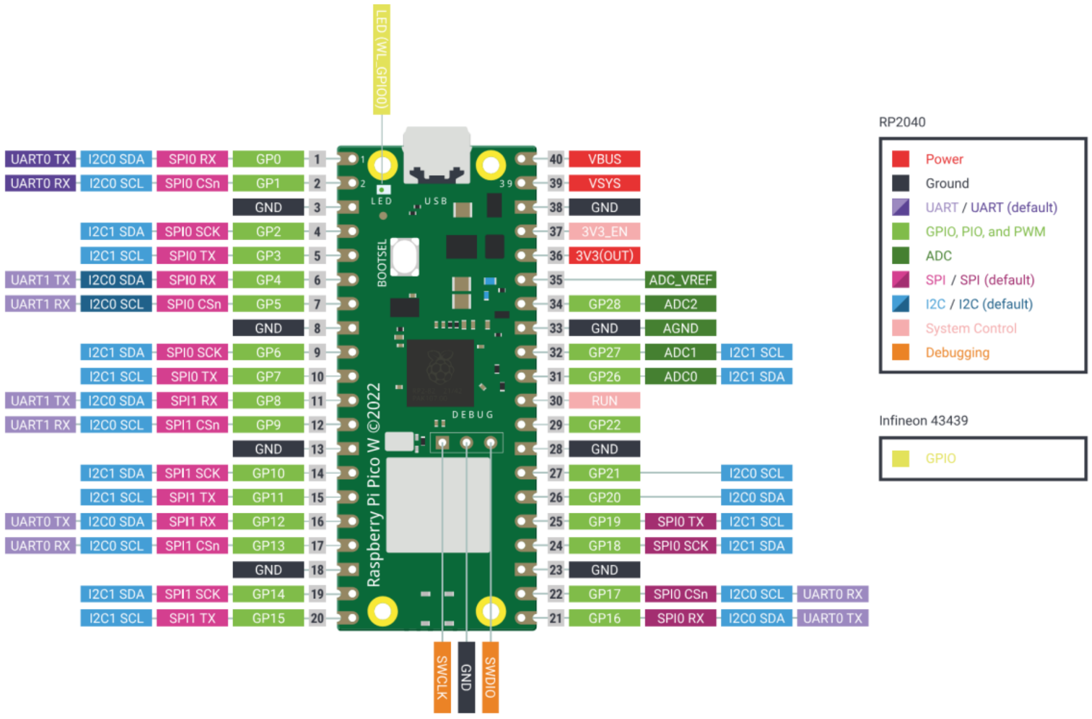
Plug the orange signal wire into GP6 on the Pico connectors. This can be adjusted in the code, however for ease of some space on the breadboard, this was selected as the default.

Finally, jump pins GP15 and 3v3V OUT between the Pico. This is to tell the program running on the Pico to use servo mode rather than differential motor control.

Wiring complete

All wires apart from power supply should now be connected on your breadboard, and you can move onto programming the Pico!

Pico Pinout:


LED (WL_GPIO0)
UART0 TX I2C0 SDA SPI0 RX GP0 1
UART0 RX I2C0 SCL SPI0 CSn GP1 2
GND 3
I2C1 SDA SPI0 SCK GP2 4
I2C1 SCL SPI0 TX GP3 5
UART1 TX I2C0 SDA SPI0 RX GP4 6
UART1 RX I2C0 SCL SPI0 CSn GP5 7
GND 8
I2C1 SDA SPI0 SCK GP6 9
I2C1 SCL SPI0 TX GP7 10
UART1 TX I2C0 SDA SPI1 RX GP8 11
UART1 RX I2C0 SCL SPI1 CSn GP9 12
GND 13
I2C1 SDA SPI1 SCK GP10 14
I2C1 SCL SPI1 TX GP11 15
UART0 TX I2C0 SDA SPI1 RX GP12 16
UART0 RX I2C0 SCL SPI1 CSn GP13 17
GND 18
I2C1 SDA SPI1 SCK GP14 19
I2C1 SCL SPI1 TX GP15 20
40 VBUS
39 VSYS
38 GND
37 3V3_EN
36 3V3(OUT)
35 ADC_VREF
34 GP28 ADC2
33 GND AGND
32 GP27 ADC1 I2C1 SCL
31 GP26 ADC0 I2C1 SDA
30 RUN
29 GP22
28 GND
27 GP21 I2C0 SCL
26 GP20 I2C0 SDA
25 GP19 SPI0 TX I2C1 SCL
24 GP18 SPI0 SCK I2C1 SDA
23 GND
22 GP17 SPI0 CSn I2C0 SCL UART0 RX
21 GP16 SPI0 RX I2C0 SDA UART0 TX
LED
USB
BOOTSEL
DEBUG
Raspberry Pi Pico W ©2022
SWCLK
GND
SWDIO
RP2040
Power
Ground
UART / UART (default)
GPIO, PIO, and PWM
ADC
SPI / SPI (default)
I2C / I2C (default)
System Control
Debugging
Infineon 43439
GPIO