ScraphEEp 2020 Technical Help

We're all from different backgrounds and all at different points in our studies, so unless you're already an EE wizard, give this a quick read through for some general advice and help. If any of it doesn't really make much sense, just ask any of the EARS Committee or academics, we'll happily explain and help.

If you have a fantastic idea for your contraption that doesn't use the provided kit, then go for it! We'd love to see some original ways of completing the challenge and getting the aliens across the finish line in imaginative ways. So if you need technical help with stuff that we haven't covered in this document, just ask and we'll do our best to help.

WiFi

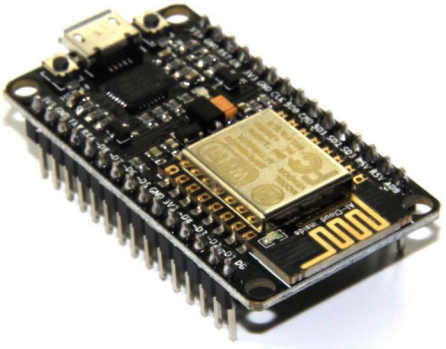
This challenge (usually) uses WiFi to communicate between your contraption and the controlling device (laptop etc). We've provided an access point in the lab for testing and the challenge. You may also wish to create your own mobile hotspot to test this - as during testing there is likely to be a lot of network traffic. Don't try to use Eduroam - it won't work apparently.

SSID: EARSAP1
Password: ears-wifi

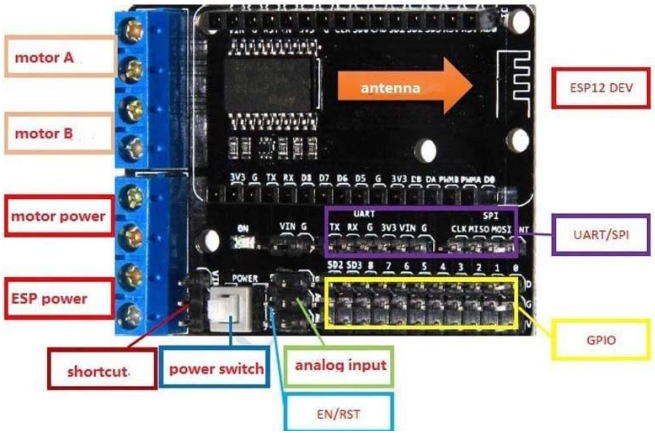
Electronics Setup

Starter Kit

1. ESP - a WiFi ready microcontroller - program this with the Arduino IDE and with the help below. The brains of your device!

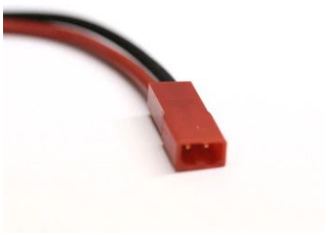


2. ESP motor shield - this allows the ESP to drive motors. The brawn of your device!

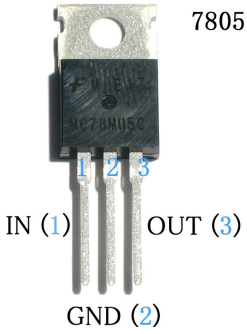


3. Servo motor - these motors will move to a specific position (very useful for robot arms etc) and have three connections (5V Red, GND Brown and Control Orange). We have a few extra of these spare, so feel free to ask for another if you want one.

4. JST Battery connector - this will connect to your LiPo battery, but only use a LiPo if EARS Committee have given you the go ahead.



5. 5V regulator. This will take in your LiPo voltage (8.4V for all your motors, VM) and bring it down to 5V (for the ESP and Servos, VIN)

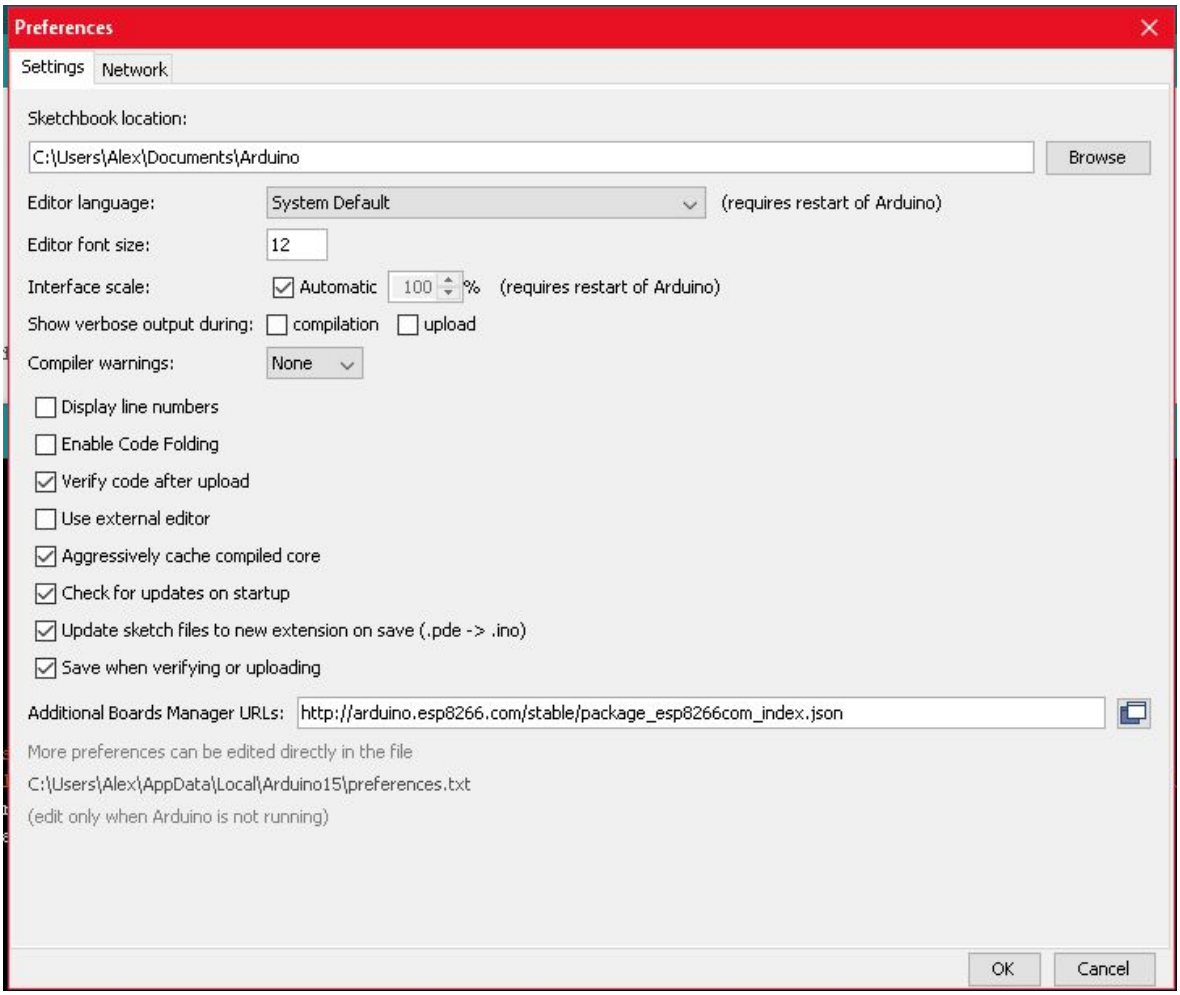


6. As well as all this very convenient hardware, we're also providing some basic code to allow you to control the ESP, and hence the motors and servo motor from your laptop. Head to https://github.com/SurreyEARS/scrapheep18/archive/master.zip to download this library and client (courtesy of the EE Wizard Phil).

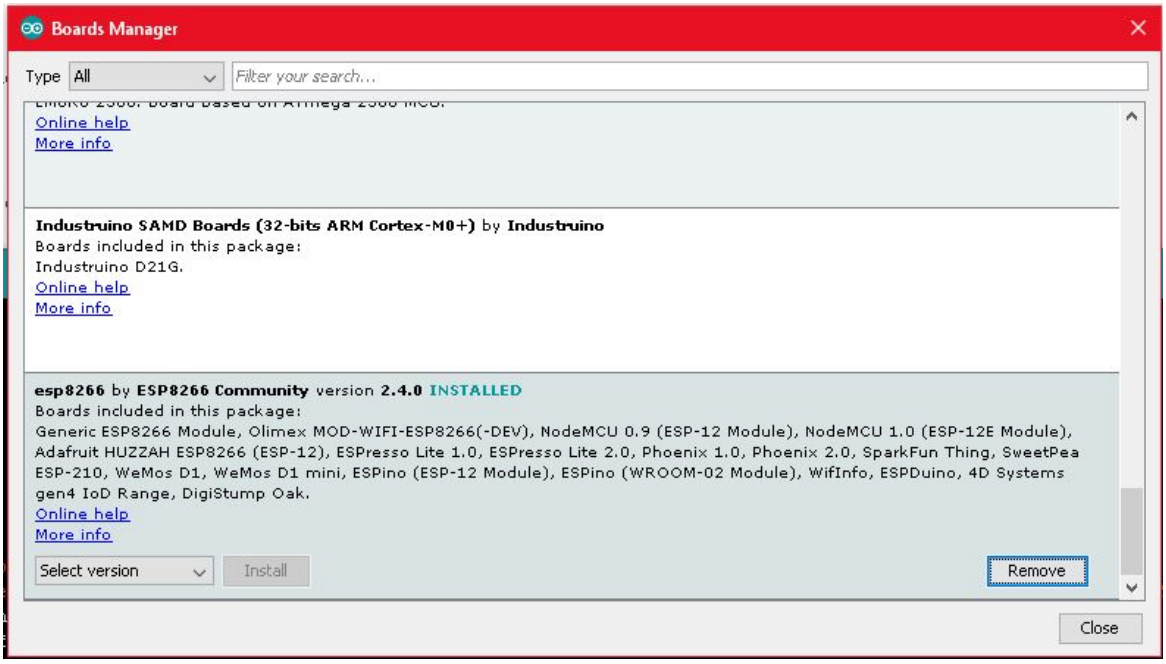
Setting Up The Arduino IDE for ScraphEEp

Some of this is confusing, but don't worry, ask any of the EARS committee at any time throughout the day and we will help you out!

1. (Recommended) Install the latest version of the Arduino Software from www.arduino.cc/en/Main/Software
2. Download the project from the link above, unzip it
3. In the Arduino IDE, navigate to **Sketch>Include Library>Add .ZIP Library** and select "lib-ScraphEEp-2K18" from the unzipped folder
4. Navigate to **File>Preferences** and set "Additional Boards Manager URLs" to **http://arduino.esp8266.com/stable/package_esp8266com_index.json** and click "OK"



5. Navigate to **Tools>Board>Boards Manager** and scroll down to find the entry for **esp8266 by ESP8266 Community**



6. Click on this entry and install the latest available version
7. To program the ESP correctly, you will need to tweak a few settings in the **Tools** menu. This will also make everything work the fastest:
    a. Board: *NODEMCU 1.0 (ESP-12E Module)* - **this should then bring up the following options (maybe)**
    b. Flash Size - *4M (1M SPIFFS)* or *4MB (FS: 2MB OTA:~1019KB)*
    c. Debug Port - *Disabled*
    d. Debug Level - *None*
    e. IwIP Variant - *v2 Prebuilt (MSS=536)* or *v2 Lower Memory*
    f. CPU Frequency *160MHz*
    g. Upload Speed - *921600*
    h. Port (select as needed when ESP is plugged in)
    i. Programmer - *AVR ISP*

    **If any of these options don't appear like you'd expect, that's okay - let us know and we can double check everything.*

ESP - Writing Your Program

1. Add the ScraphEEp2k18 library to your code: **#include <scrapheep.h>**
2. (If you run into problems compiling this such as *" multiple libraries were found for "WiFiUdp.h" "* try adding **#include <ESP8266WiFi.h>** above this)
3. Above the line "void setup()", type "ESPControl control;" to create the ESP manager.
4. Once this is done, simply call "control.init();" inside setup() to connect your processor to the WiFi.
5. Finally, to start receiving data from the controller, put this code inside the loop() function:
   "uint8_t *data = control.processPacket();
   If (!data) return;"

6. All the data from the control software will be inside the data[] array, so simply access the data by calling data[the data ID].
7. The data IDs are as follows:
    a. MOTOR_A
    b. MOTOR_B
    c. C1
    d. C2
    e. C3
    f. C4
8. **Alternatively, to save time doing this, you can work straight from Phil's example by clicking "File->Examples->ScraphEEp-2K18->PhilsRobot".**

ESP - Interfacing and Use

The ESP and motor shield as given to you have:

- Two separate power supplies (*VIN* for the ESP itself, and *VM* for the motor driver output) on the blue screw terminals at the end -NOTE: **VIN<9V and VM<36V**
- Two two-terminal screw connections for motors on the blue screw terminals, labelled A and B.
- Nine Digital Pins labelled D0-D8 and set up to plug directly into a servo (see Technical Hints) - **NOTE: D1-D4 pins are shared with the motor driver chip, so can't be used as logic inputs/outputs unless the motor driver isn't in use**

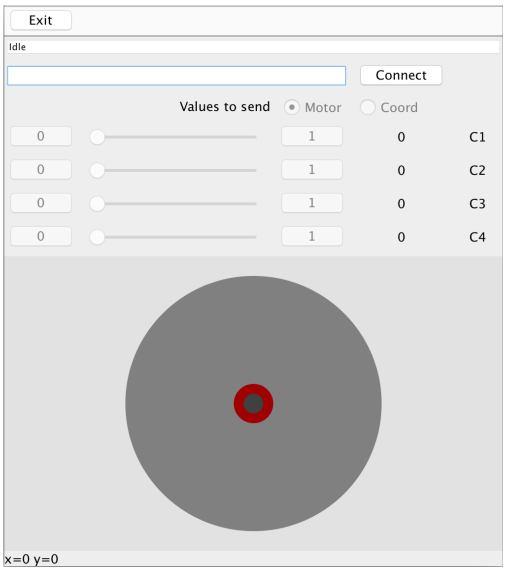
To Power the ESP and the motor shield you have a few options:

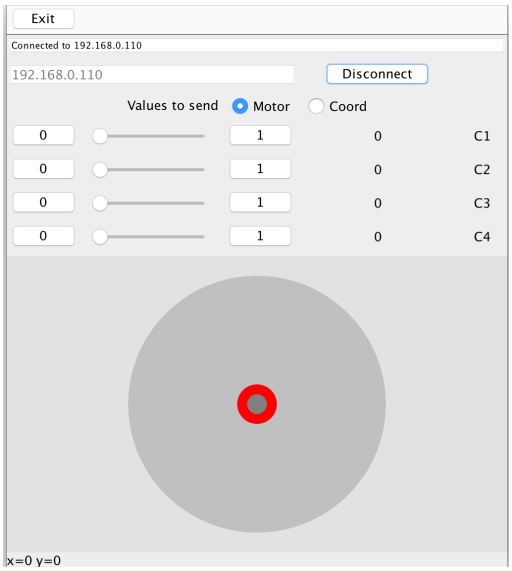
- **Single battery power** for both VIN and VM - the easiest way to power everything. Place the pin bridge over the pins directly behind the blue screw terminals to connect VIN and VM together on the board. This is easier, but we prefer the following option.
- **Single battery power with 5V regulated input** for VIN (as on Alex's demo). Use the 7805 5V regulator in your starter kit and connect it heatsink-side-up in the three neighbouring screw terminals marked VM G VIN (you might also want to check a datasheet online). Feed your motor power to the VM and G terminals, and the ESP and servo power pins will be powered off 5V. This is really useful for stopping servo motors from blowing up if you want to use them

If using the LiPo battery, **please respect the battery**. We will only be giving out batteries to teams who can demonstrate that their system is not short circuiting (to prevent any explosions etc), so before you get your battery it's best to instead power your systems from a bench power supply to test everything (the battery voltage will be 8.4V full, 7.4V minimum usable)

Client

This is a Java program (created by Phil and previous committee) that we are shamelessly reusing, because it's fantastic. It allows you to send various control signals to your ESP.





1. Enter your ESP's IP address
   Written on a paper slip, or 192.168.0.1x where x is your team number *(if <10, add leading 0)*
   This can be found out by looking at the Arduino IDE Serial monitor when running the example sketch in the scraphEEp2k18 library mentioned above.
2. Click *Connect*
3. 2 control types
   a. *Motor*: send differential motor values based on joystick location
   b. *Coord*: send joystick coordinates
4. 4 control sliders (*C1-4*)
   a. Customisable, sends a value between 0 and 255
   b. 0 and 1 buttons jump to the end of the slider
5. Joystick canvas
   a. For controlling motors
6. Click *Disconnect* to close connection, *Exit* to disconnect and close application

Technical Hints

Batteries

Provided your circuit doesn't have any shorts and is not likely to develop any whilst in use, we will supply you with a 2-cell Lithium-Polymer battery which provides a high current (enough to not be an issue) at 7.4V - this will increase to 8.4V for a fully charged battery.

You don’t have to use this battery - for example, you might want to try and power a computer fan. These typically require a much higher voltage (ie. 18-24V) but a much lower current, which can be provided by stringing a few 9V batteries together in series. If you ask nicely Laurence (1st Year Lab Tech) might lend you some batteries.

However, don’t forget that the ESP’s input voltage (VIN) must not exceed 9V or things will start to go pop… And we don’t have any ESP’s spare.

In this case, you might want to consider giving the ESP a separate supply or use a voltage regulator (the 7805 regulators we provide are rated for up to 35V input voltage)

Motor Selection

We have a lot of motors that can be used - there are some salvaged DC motors in the Makerspace, some really tiny useless little ones, and some geared DC motors with yellow gearboxes (these seem pretty decent - but we haven’t tried all the salvaged ones).

If you use the yellow geared motors, be aware that there are THREE TYPES with different speeds - we aren’t really entirely sure why…

- White top, one pink shaft
- White top, two white shafts
- Black top, two white shafts

We advise that you check both your motors match before you glue them to anything (unless you want to be stuck going in circles).

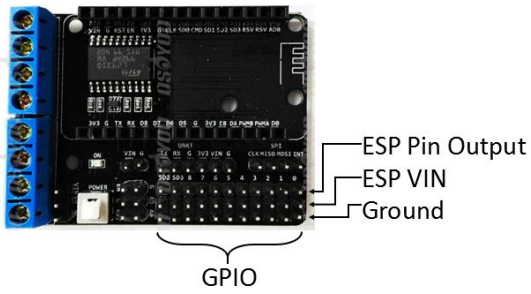
Motor Driver

The ESP motor shield that we’ve provided uses an L293DD H-Bridge motor driver chip, which can drive two motors. However, power-wise it’s not the best, it’s rated at 0.6A per channel with a peak non-repetitive current of 1.2A. From our tests, you can put a lot more than this through it for very short periods, but we don’t recommend this at all. For the yellow geared motors, that’ll probably be fine but if you want to drive a chunky motor you’ll need to figure something else out.

My Motor Draws Too Much Current!

Please please please check how much current the motor needs with a bench supply before you try and use the motor driver! We don’t have spares. If you find that you’re drawing too much current (consistently more than 0.6A), the easiest (but not elegant) way to solve the problem is to reduce the voltage across the motor by putting some diodes on the positive supply line between the battery and the motor driver (this also works for the Vm input to the ESP motor shield). We have some chunky 1N5406 diodes in the makerspace, which can each handle 3A and will probably drop just over 1V when you pass a high current through them. Ask any EARS Committee if in doubt about this.

Driving Servo Motors



We have given each team a small servo motor. They are small, so don't expect them to have loads of strength.

If your team thinks two servos could be useful, let the EARS committee know and we can find you another.

These can be used by connecting the servo directly to the ESP motor shield, which has conveniently located pins that allow us to do this. The pins provided match the pin ordering on the end of a servo motor lead.

These servo motors are designed to run off 5V, but used like this they will be running from VIN - which is only 5V if the provided 5V regulator is being used. If not regulated then the servo may be damaged.

To control the servos in software, the easiest way is to use the Arduino servo library which comes preinstalled in the Arduino IDE. Look at the PhilsRobot example for a possible way to do this.

Driving other stuff

The pins referenced in the "Driving Servo Motors" section can also be used as standard IO pins like an Arduino and can be controlled using digitalWrite and analogWrite - which produces PWM waveforms.

Apparently, PWM works on all of the ESP's pins and has a maximum value of 1023 (as opposed to 255 on an Arduino). We haven't actually tested this thoroughly so let us know if you find otherwise…