RoboWars

Preliminary
Please read through this document, follow the steps and if you have any questions feel free to ask those around you or any of the more experienced Robotics Club members.

The purpose of RoboWars, BattleBots etc is to create robots that are engaging and fun to develop & drive. Teaching you how to rapidly prototype, problem solve and drive the robots. Once progressed, you will have hopefully developed the skills to actively and meaningful engage with the Flipper Bot and the wider RoboCup competition.

RoboWars is designed to create solvable problems and space to develop creatively with mentorship. You will face challenges and that's intentional.

Priorities:
- Development for the RoboCup competition not RoboWars
- Engage students
- A stepping stone from basic knowledge to meaningful engagement
- Develop driving skills

Stages
There is an order of progression we would like all new students to follow.

1. Spend a maximum of 3 sessions driving around BattleBots.
2. Work on the test bed (see introduction document here.)
3. Build your own BattleBot
4. We want you testing, developing and competing with your robot.
5. After you've progressed and developed your skills we want you working on the proper robots.

Contents

Quick start

- Turn computer on
- Plug robot in
- Make sure battery is charged **and has a beeper attached**
- Follow Coding, general boot instructions

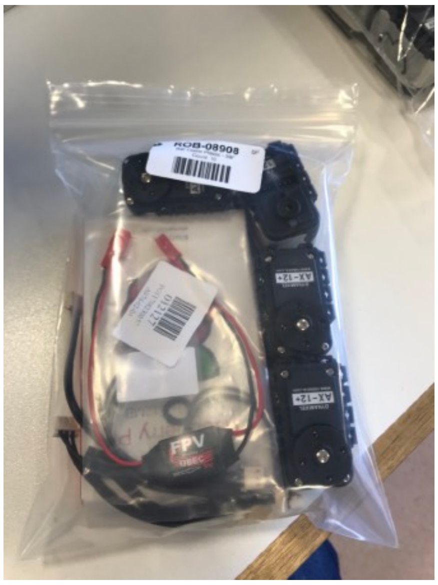
Helpful links

CCGS Robotics - GitHub
Thingiverse - CAD CAM documents
BattleBot Code - Github, currently on Ben Lane's Github

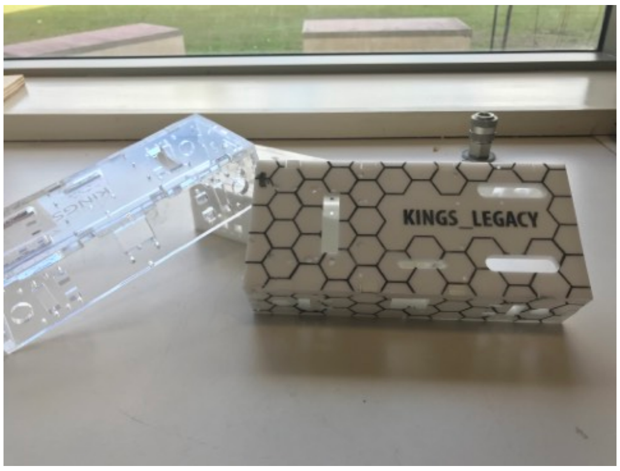
Building

Step 1.

From the blue box, labelled 'RoboWars' collect a chassis and a starting pack that looks like this (keeping in mind they will not look identical to the images):



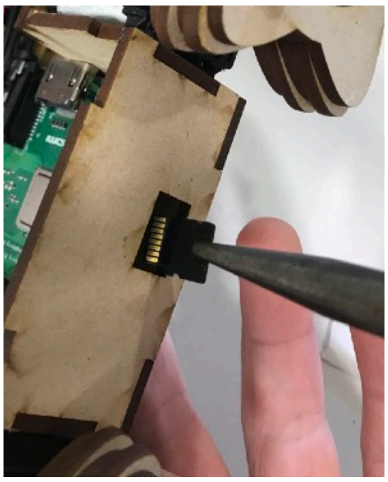
And a chasis:



Or alternatively a MDF version specifically designed for RoboWars.

Step 2.

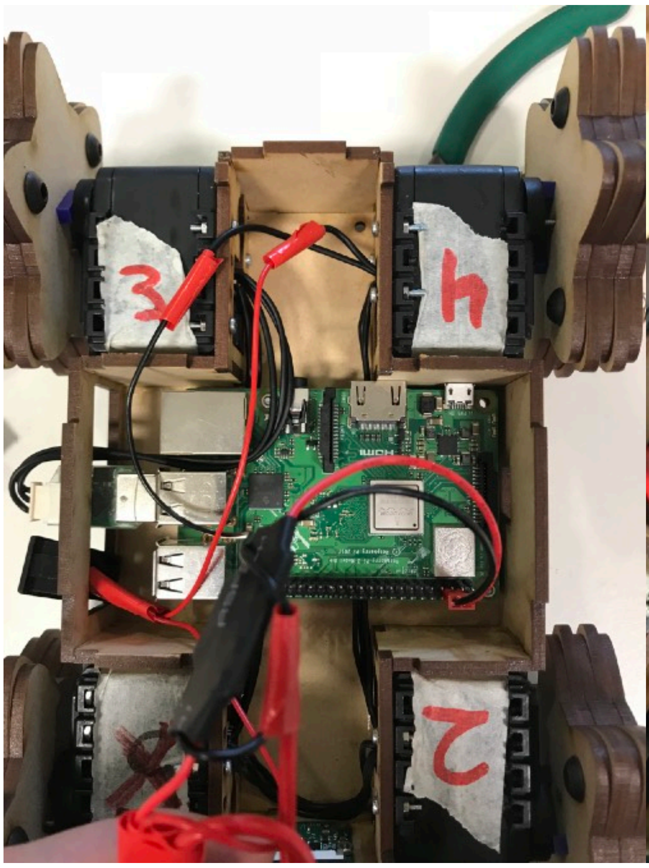
Get components from the starting pack (wiring harness, Dynamixel cables (3 piece cables with cream ends), raspberry Pi etc) and screws, attaching like the following images. If you get stuck, look at other robots, they are all very similar.

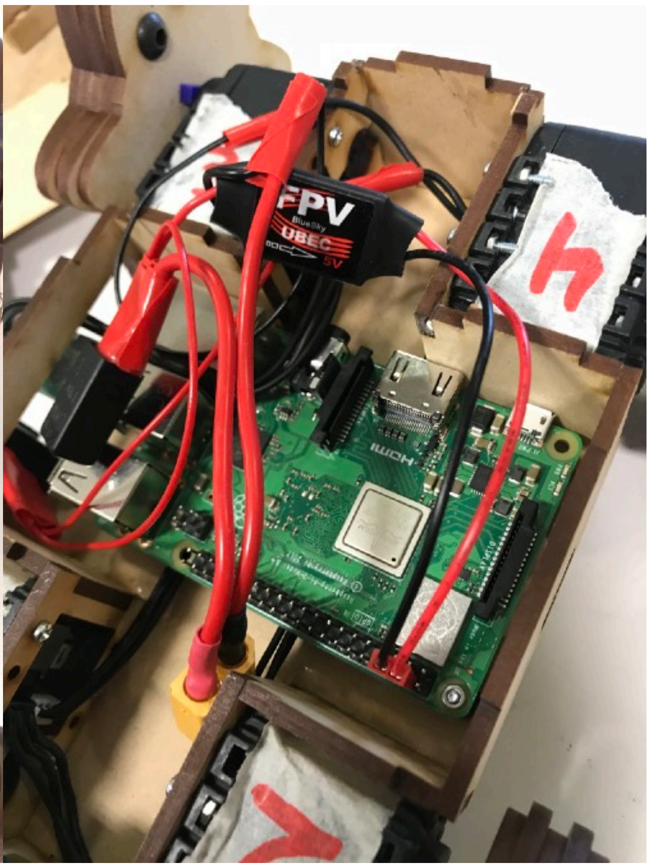


Things to note: the Dynamixel cables need to be daisy chained, make sure you get the pins right on the Raspberry Pi and be very careful with the SD card.

The battery must be plugged in last. Make sure you also have a beeper with it, we will be annoyed if we find there isn't a beeper attached as this can easily cause a dead expensive battery. **If mishandled these batteries are a bomb.**

Helpful documentation can be found here.





Helpful building tips

MDF

For the cutting out of the MDF parts:

1. Ask a responsible staff member to cut 2 pieces of Medium Density Fibreboard (MDF), each piece having dimensions of 800mm x 400mm.

2. Ask a teacher to help you cut out a copy of the 2 attached nested files onto the MDF, using the laser cutter. Here and Here.

Raspberry Pi

- The Raspberry Pi only needs two screws in opposite corners.
- A very good cable management idea to put your cables underneath the Raspberry Pi where possible.
- Also it makes it easier if the USB ports on the Raspberry Pi can be easily accessed.

Screwing

- We use 2mm screws for the servos.
- When screwing make sure there is a nut in the servo screw spot otherwise the screw will just fall out.
- Make sure that the screws are not bending the MDF.

MDF chasis construction

- It’s a good idea to hot-glue the 2 pieces that touch the servo together. This will make final construction much easier.
- Before gluing, put assemble the robot and check to see if everything fits together.
- Do not glue the bottom piece of the chassis as this is used to access the Pi and servos.

Wiring

The wiring diagram can be found here.

Coding

You can find a basic code structure similar to the Test Bed on GitHub or here. Use this to first test an individual servo and then change the code to gain full control of the robot.

General boot instructions to run/drive the BattleBot

The process to connect to the BattleBots is being reviewed to make them more streamline. This documentation may change.

The following instructions are Ben Lane's notes from GitHub

1. Ask a mentor for a battery to power the BattleBot. Plug a beeper into the battery, to ensure that you won't drain the battery too much, killing it. Plug the battery into the connecter in the robot, and ensure that the Raspberry Pi has a red light turned on, and a flickering green light.

2. The robot emits a wi-fi network under the name 'kings_legacy.battlebot'. Connect to this wifi network with a robotics laptop, using the password 'raspberry' when required.

3. Once connected, open a new terminal window and enter the command 'ssh pi@192.168.100.1' Enter the password 'raspberry' when asked. This command connects the two devices though a ssh connection, allowing the user on the laptop to control the raspberry pi though terminal commands.

4. Enter the command 'cd Documents/BattleBot'. This tells the raspberry pi what folder we will be running our code from.

5. Enter the command 'python3 run.py'. This runs the python TCP server that will receive servo commands from the laptop, and execute them with functions defined in functions.py

6. Ask a mentor for a gamepad. We use basic Logitech controllers (gamepads) to control our robots. Plug the gamepad into a usb port on the robotics laptop.

7. Leaving the current terminal window open, find the BattleBot folder on the desktop of the laptop. Open this and open the python file 'client.py'. Run this file in IDLE.

8. You're good to go!

Camera feed

1. Open a new terminal window and run the command 'cd Desktop/Battlebots'. This will tell the laptop to look for software inside the 'battlebots' folder on our desktop.

2. Run the command 'bash camera.computer.bash' This will set up the laptop to begin receiving data from the raspberry pi camera feed.

3. Repeat steps 3 and 4 from above.

4. Enter the command 'bash camera.pi.bash'. This will tell the raspberry pi to receive data from the picamera and send it to the laptop.

5. The laptop terminal window should now be showing the cache filling up. This may take a while, as we try to run the camera on a fine balance of resolution vs frame rate.

6. Once the camera feed appears, you should see the point-of-view of the BattleBot. Attach the bottom of the robot by clipping it into place and securing it with masking tape.

7. Test out your new robot on the simulated terrain in the Design and Technology building!

Important notes
Below will be some important tips to remember

Raspberry Pis and 2ax (usb -> servo adapter)
It is very important that you don't break anything! If you do, the world will not crumble down, just tell us. The **most valuable component is the 2ax**, protect it at all costs.

How you protect electronic components from their biggest enemy, static, is by not putting them on surfaces. To rephrase, make sure there is something in between the (table) surface and the Raspberry Pi. Preferably a static proof bag (the bluey grey one) or the box it came in.

Servos
There are two types of servos that you will be interacting with: AX-12**A** and AX-12**+** they look exactly the same but slightly different dimensions so wont fit into the same holes, therefore the same chassis. The old acrylic chassis use 12A, the MDF chassis will use 12+ until we use all of the 12+ up.

The second note with servos is they need to have the correct Servo ID. You'll notice they have a number written on them, this needs to be the correct number 1-4 in the correct spot on the robot.

When you figure out why, with detail, please tell me (the past student, Felix) as this is an important concept to learn. You can look at the other robots to figure out the correct order.

The screws we use for servos are 2mm.

Setting Servo IDs
Now how to set the Servo IDs. This is copied straight from the User Documentation and Troubleshooting Guide, a document that as time goes on you should become familiar with.

1. Navigate to BattleBot/Dynamixel-Utilities
2. Run the file dxl_utils.py in IDLE3, go to Run > Run Module
3. Follow the prompts

Difference in chassis
It was briefly mentioned that different servos go with different chassis, this is important to remember. Another important note is that the old acrylic chassis offer more customisation as they have more flexibility with space, the MDF are very efficiently designed so have significantly less space. This is your choice to make.

Missing parts
You may notice that information has been left out of this manual. This is intentional, we want you to be asking questions, doing your own research and figuring out things for yourself as when you progress to the proper competition, that's what you'll be doing a lot of.

It's ok to look at other code (that includes the clubs own GitHub repositories) and copy the ideas in that code so long as you credit the original author. The competition is open source so it's ok to do the same thing in the competition.

Feedback
We want your opinions on the whole experience to make the development, the robots and Robotics as great as possible. If you've got some thoughts/opinions feel free to send the team an email or tell us.

Testing and bug finding

Often things don't work how we want them to or not at all. For reliable functional robots we can't have this. The solution is bug finding and testing and testing and testing our robots.

Bug finding

A good process to bug find is isolating the problem. Find out what you can confirm works and then by a process of elimination work backwards to find what isn't working.

I.e. if a servo isn't working because the servo cable is broke.
- Test each servo one by one
- For the servo that isn't working, switch the cable out, problem found.

Testing

When you first get your robot working its such a great feeling! Congratulations!

Before you get it driving on the terrain make sure its working how you expect it to in the club room. Try moving individual servos, find what its rough limits are etc.

Reliability testing

To make the best robot you can make it needs to consistently work well, we need it to be reliable. To do this slowly incrementally push and push the robot in more and more challenging environments and situations while using common sense.

I.e. what happens if I can't see the robot? What happens if the controller cable get caught? What happens if the controller cable gets unplugged?

Track time

We want you out on the track! We want reliable, robust robots; that means you getting your robot out driving!

In the RoboCup competition you get points by, among other ways, driving your robot as fast as you can, end to end on a section of track in 5 minutes. We want you doing this!

When there are other BattleBots around you may have a RoboWars competition. The exact competition hasn't been finalised so we'd love your feedback on this.

Optimising

After the initial enjoyment you may have been asking yourself a few questions: Why is it getting stuck on the terrain? Can I make the wheels bigger? Would it be good to have an arm like the proper robots?

These are questions we want you to be asking! …and solving, optimising, making better robots in a process called rapid prototyping. This is a fantastic skill that can then be applied to the actual RoboCup competition.

Please ask before 3D printing as similar parts may exists (like arms) or your design may use too much 3D printing material.

An example of improvements could be wheels, here are some examples of versions we've tried in the past. This includes tracks. You can find these in the RoboWars box, please use these for inspiration.