INF3 System Design Project

Robot Football

User Guide — Group 9E

Ramsey El-Naggar (s1340038)

Jake Greenshields (s1572908)

Naohiro Kakimura (s1343880)

Andrew Smith (s1326224)

April 2016

Contents

1 Introduction **1**

2 Setup **1**
2.1 Equipment list . . . 1
2.2 DICE installation . . . 1
2.3 RF configuration . . . 2
2.4 Programming the robot . . . 2

3 Robot **2**
3.1 Structural overview . . . 2
3.2 Using the robot . . . 3

4 Gameplay **4**
4.1 Running specific planning subtasks . . . 4
4.2 Running the main strategy . . . 4
4.3 Vision calibration . . . 5

5 Troubleshooting **7**
5.1 Vision feed is blue/black . . . 7
5.2 Vision feed will not close . . . 7
5.3 Blocked USB port . . . 7
5.4 Robot struggling to move or turn . . . 7

A RF configuration guide **8**
A.1 Configuring the SRF stick . . . 8
A.2 Configuring the Arduino . . . 8

B Arduino programming guide **9**
B.1 Launching the Arduino IDE . . . 9
B.2 Programming the Arduino . . . 9

List of Figures

1 Gertrude is pleased to meet you . . . 2
2 Changing the battery pack . . . 3
3 Configuration file format . . . 4
4 Example top-plate (blue team, green unique colour) . . . 5
5 GUI colour calibration panel . . . 6
6 GUI status panel . . . 6
7 GUI tracker panel . . . 6
8 GUI settings panel . . . 6
9 The Micro USB port on the Arduino . . . 9

List of Tables

1 List of remote-control commands . . . 4
2 List of terminal commands . . . 4
3 Meaning of sensor messages . . . 4
4 List of strategy commands . . . 5

1 Introduction

This document explains how to use the system created by Group 9E for the System Design Project 2015-2016, a third year course within the School of Informatics, the University of Edinburgh. The aim of the project was to design, construct and program a LEGO robot to play robot football. The purpose of this document is to allow the reader to fully understand how to use both the code and the robot to get the robot playing football, as per the SDP 2015-2016 rules. Throughout this document, it is assumed that the reader has the knowledge and experience at the level of a first year Informatics student.

The vision and strategy sub-systems were created collaboratively by Group 9 and Group 10, which together form Team E. Section 4 assumes that Group 10's robot is operating at the same time (a requirement for the system to play football properly, as robots are assigned specific roles, with Group 9 as the attacker and Group 10 as the defender), but this document explains only the knowledge required to operate Group 9's robot.

This document is structured as follows: Section 2 explains the user requirements and how to install the code and configure the equipment; Section 3 gives an overview of the robot sufficient to operate the robot by itself; Section 4 explains how to operate the full system in order to play a match; Section 5 gives solutions to known problems that have been encountered when operating the full system.

2 Setup

2.1 Equipment list

The following equipment is required to allow the robot to play football:

- one standard DICE machine (connected to the vision feed for the desired pitch room)
- one Ciseco 858-915MHz SRF stick
- the robot itself, including the detachable top-plate
- eight 1.2V AA batteries
- one AA battery charger
- one tool to measure the charge of the 1.2V AA batteries
- one USB to Micro USB cable (only required to program the robot)

2.2 DICE installation

This section describes the steps required to download and install the the system onto a standard DICE machine.

First, the main Git repository for the project must be cloned. This is done by opening a terminal window and entering the following command:

```
$ git clone https://github.com/Ramsey144/SDP-2016-9E.git
```

The vision repository (which was shared with Group 10) is linked to the main repository as a Git submodule. This must be initialised by navigating to the root of the main repository and executing the following commands:

```
$ git submodule init
$ git submodule update
```

```
$ cd python/vision     // this is the path to the submodule
$ git checkout master
$ git pull
```

The system should now be installed and ready to use.

2.3 RF configuration

In order for the machine that is running the system to send commands to the robot, the SRF stick and the RF link on the Arduino must be configured to work together. The rest of this document assumes that this configuration has already taken place. For completeness, a guide on how to carry out this configuration is included in Appendix A.

2.4 Programming the robot

The rest of this document assumes the Arduino inside the robot is already programmed with the code required to play football. For completeness, a guide on how to program the Arduino with the correct code is included in Appendix B.

3 Robot

3.1 Structural overview

The robot has a cuboid shape, with two small ball bearings at the front and two large wheels at the back. The robot moves forwards and backwards by turning the wheels in the same direction. The robot rotates clockwise and anticlockwise by turning the wheels in opposite directions. There are two slightly curved grabbers fixed onto the front of the robot that can be used to grab the ball. Once grabbed, the ball is kept in a specific place, held in this place by 'keepers' which stick out slightly and ensure the ball remains in position. There is a spinning kicker located inside the robot, which can be rotated around to kick the ball forwards in a straight line.

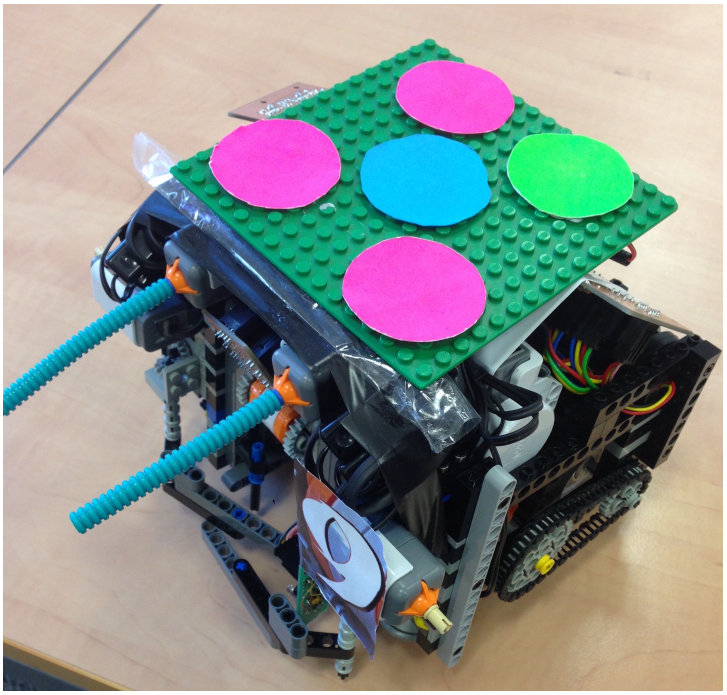

Figure 1: Gertrude is pleased to meet you

Each wheel and the spinning kicker are powered by three separate LEGO NXT motors, and can be turned independently. Each grabber arm is powered by a 9V LEGO Electric Technic Mini-Motor, but the two motors are connected to the same Arduino port, meaning the grabbers move together in a grabbing or ungrabbing motion.

An infrared (IR) sensor is attached to the left side of the 'keepers'. The sensor is placed such that when the ball is in the grabbed position, the sensor will give a distinctly different reading from when the ball is not grabbed. There are four LEGO NXT touch sensors that detect when the robot collides with a wall or another robot whilst moving forward or rotating. Two are located at the front and one is located on the left and right.

To increase the smoothness of the ungrabbing motion, two LEGO Technic touch sensor are located behind each grabber such that the sensor will be pressed when its grabber arm opens. To help the robot move in a consistently straight line, and to allow more precise movement commands to be executed by the robot, the built-in rotary encoders within the wheel-motors are connected to the Arduino; similarly for the kicker-motor.

3.2 Using the robot

Switching on the robot

The robot is turned on by plugging in the battery pack. This consists of eight 1.2V batteries. The battery pack can be accessed and swapped from the back of the robot, as shown in Figure 2. The Arduino contains a reset button, but this is not used during the normal operation of the robot.

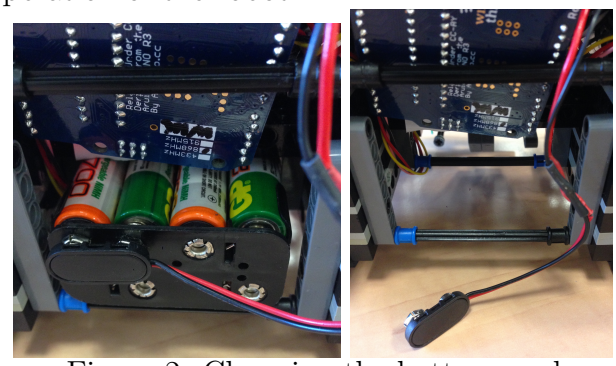

Figure 2: Changing the battery pack

In order for the kicker to function as intended, the kicker should be in the fully unkicked position before the robot is switched on. The robot then takes around three seconds to configure the IR sensor, after which the grabbers will close into their default grabbed position, indicating the robot is ready to be controlled.

The batteries should always be as fully charged as possible when operating the robot, particularly before entering a match. The robot will function as expected for the duration of two full matches in a row, but may not operate as expected after this. To ensure the batteries are fully charged, place each battery in the battery charger until the charger indicates the battery is fully charged. Then double-check this is the case by using the battery test tool. If a battery is not 100% charged, charge the battery again. If the battery is less than 70% charged after charging it twice in a row, it is considered 'dead' and a replacement battery found.

Manual control

After switching the robot on as described above, the robot can be remote controlled by plugging the SRF stick into the USB port of the DICE machine and running the following terminal command:

```
./[project-root-directory]/python/remote-control-group09 X
```

where X is the port number for the SRF stick (port numbers are assigned by the order in which USB devices are plugged in, starting with 0).

The complete list of keyboard commands accepted by the remote-control application are given in Tables 1 and 2. Where 'X' occurs, this represents the need for a three-digit number to quantify the particular action - note that this must be a three-digit number - use leading 0s if the number is less then 100. The meanings of 'messages' that can be sent by the robot and displayed in the remote-control application are given in Table 3.

Note that the robot will automatically grab the ball if the ball is detected by the IR sensor. Also, the robot will automatically halt if a collision is detected. Additionally, the robot automatically ungrabs just before kicking, and automatically resets the kicker and ungrabs after kicking.

Command	Action
tX	Move forward X rotations
gX	Move backward X rotations
hX	Rotate clockwise X rotations
fX	Rotate anticlockwise X rotations
w	Move forward
s	Move backward
d	Rotate clockwise
a	Rotate anticlockwise
SPACE	Halt movement and rotation
bX	Kick with X power ($X \in [0, 100]$)
k	Kick with full power
i	Kick backwards (reset kicker)
j	Grab ball (close grabbers)
l	Open grabbers

Table 1: List of remote-control commands

Command	Action
q	Quit the display
MINUS	Clear the display
BACKTICK	Freeze/resume the display

Table 2: List of terminal commands

Message	Meaning
*	The movement/rotation is complete
B	The ball has been detected
G	The ball is no longer detected
F	One of the front bumpers has been pushed
L	The left bumper has been pushed
R	The right bumper has been pushed

Table 3: Meaning of sensor messages

4 Gameplay

4.1 Running specific planning subtasks

In order to test individual subtasks, (e.g. 'acquire ball' and 'turn to ally') the system can be run with a 'mock strategy' that executes subtasks depending on which key the user presses. As this is not required to play a match, the procedure to carry out these tests is not explained in this document, but involves running the full system as described in Section 4.2 with 'None' for the argument in the configuration file relating to the SRF stick for the team which is not being tested.

4.2 Running the main strategy

The full system is launched during a match via the following procedure:

- Insert the SRF sticks for both robots.
- Place both robots on the pitch, and switch them both on (as described in Section 3.2).
- Create a configuration text file with the format shown in Figure 3 (one argument per line, no newline at the end of the file). The meaning of the arguments is given here:

```
[video size]
[pitch number]
[team colour]
[group 9 unique colour]
[group 10 unique colour]
[group 9 SRF stick USB port number]
[group 10 SRF stick USB port number]
```

Figure 3: Configuration file format

video size ('big' or 'small') This depends on the particular camera equipment connected to the machine in use. The argument refers to the range of possible numbers for the camera settings.
pitch number (0 or 1) 0 for the pitch room closest to the main SDP lab.
team colour ('blue' or 'yellow') The colour in the center of the top-plate.
unique colour ('green' or 'pink') The unique colour on the bottom left of the top-plate.

- Execute the following terminal command:

```
$ ./[project-root-directory]/python/launch [path/to/configuration/file]
```

- Calibrate the vision system until all four robots and the ball are reliably detected (the procedure for vision calibration is described in Section 4.3).
- The system should then launch. The game is played by inputting game commands as per the rules and referee decisions. A full list of commands accepted by the system is given in Table 4.3 (along with the terminal commands given in Table 2).

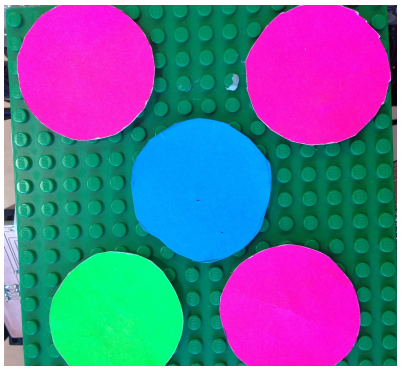

Figure 4: Example top-plate (blue team, green unique colour)

Note that the system can be launched without the configuration file, in which case an interactive menu will ask the user for the required information. This method of launching the system is inconvenient during matches however.

The system copies everything printed to the application's display (including error messages) to a log file within `[project-root-directory]/python/`. Note that when the display is 'frozen' via the BACKTICK command, logging continues.

The meaning of the strategy command actions is given here:

Play ball means jumping straight to executing the main strategy (no kick off or penalty).

Swap sides means swapping which side is considered the defending side, and which side is considered the attacking side (this only needs to be done if the defending/attacking side is mistakenly entered at another point).

Reconnect means attempting to reconnect with a robot (this only needs to be done if a connection fails, for which the user is given a warning)

For actions that require further input (e.g. qualifying whether a penalty is being defended or attacked, which robot is taking or defending the penalty, which side is being defended), the system will ask the user for all required input.

4.3 Vision calibration

Command	Action
SPACE	Halt both robots
b	Play ball
k	Kick-off
p	Penalty
HASH	Swap sides
c	Reconnect

Table 4: List of strategy commands

Vision calibration settings are saved on each machine whenever the user changes these settings. As calibrations have been carried out on every machine in the pitch rooms, this should not need to be done again. For completeness however (and because of the volatility of the vision feed), the procedure required to adjust the vision settings is given here:

- Firstly, calibrate the colours for robot and ball detection using the GUI colour calibration panel (Figure 5). This is done by setting the minimum and maximum hue, saturation and value values for each of the five colours - blue, yellow, red, green and pink. The track-bars for minimum values should always be less than the corresponding maximum value. A larger range between minimum and maximum values makes tracking more stable, but increases the chance of false positive detections. The pink and red colours in particular are often mistaken for each other. Colour calibrations can be saved by pressing 'Write Configurations' or reloaded from the previous save by pressing 'Reload from File'. It may be useful to uncheck 'Show Robots' and 'Show Ball' in the GUI settings panel (Figure 8) so that the contours for each colour can be seen clearly.

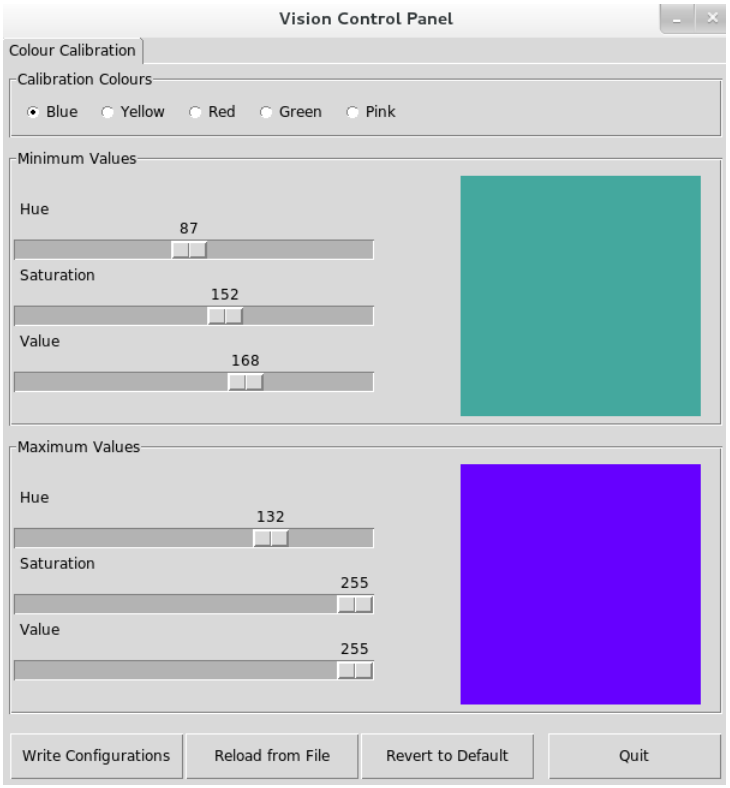

Figure 5: GUI colour calibration panel

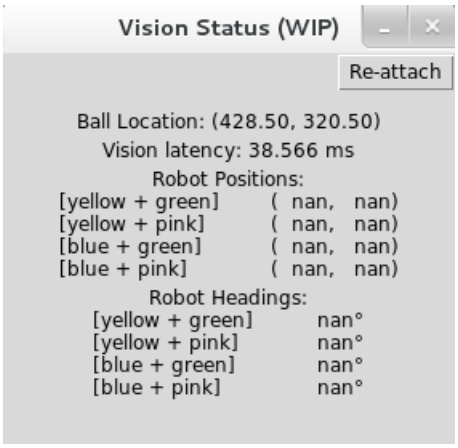

Figure 6: GUI status panel

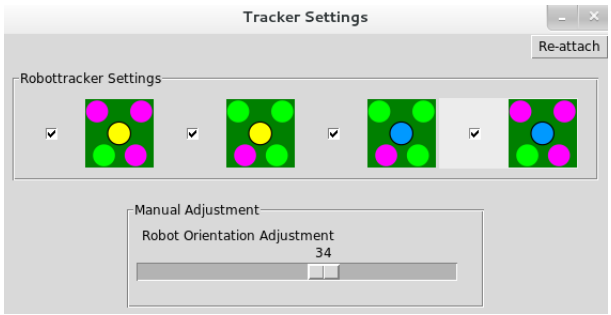

Figure 7: GUI tracker panel

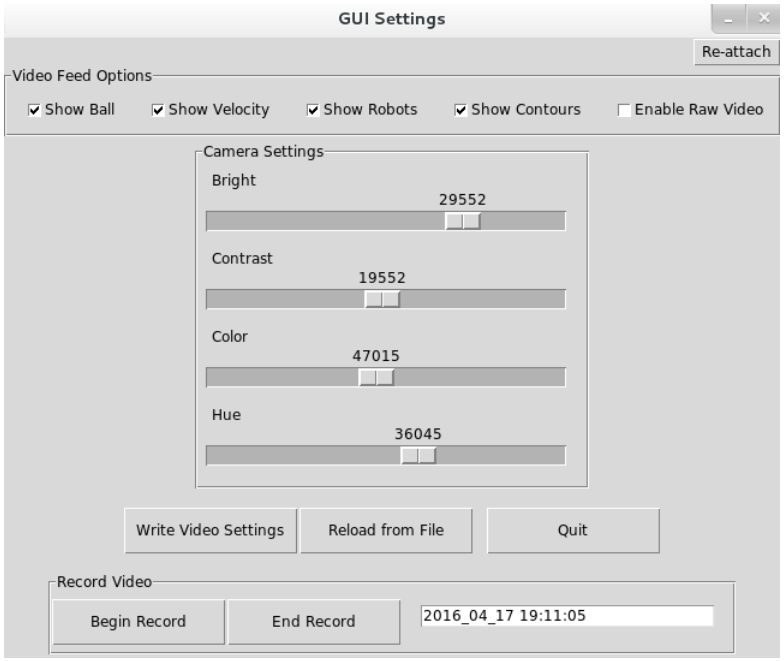

Figure 8: GUI settings panel

- If calibration is proving difficult it may be necessary to adjust the camera settings for brightness, contrast, colour and hue. Track-bars to adjust these settings are located within the GUI Settings panel (Figure 8). The main purpose of changing these settings is to ensure that different colours on the pitch actually appear as different colours on the vision feed. As with the colour calibration settings, the camera settings can be saved or reloaded.
- The final step is to account for any systematic error in the robot orientation calculations by adjusting the 'Robot Orientation Adjustment' track-bar in the GUI tracker panel (Figure 7).
- Throughout calibration, it is important to ensure that only the robots that are seen on the vision feed are being tracked. This can be done by selecting the robots to track in the GUI tracker panel (Figure 7). If the selections are not accurate, robots may be identified incorrectly.
- To see the raw vision feed before transformations and cropping, check 'Enable Raw Video' in the GUI settings panel (Figure 8).

5 Troubleshooting

This section outlines the solutions to known problems that have been encountered when operating the full system.

5.1 Vision feed is blue/black

Open a terminal and type `xawtv`. This should open the raw vision feed in a separate window. Right-click on the window and make sure that 'input' is set to 'S-Video' and 'TV norm' is set to 'PAL'.

If the vision feed remains blue/black, restart the terminal. If the problem persists, restart the machine. If the problem still persists, consult with the course technician.

5.2 Vision feed will not close

Occasionally, when the system is exited, the vision feed window will freeze and remain open. This is a bug, and the only solution is to force quit the window. This problem likely occurs due to a thread in the application failing to terminate, although the reason this only happens sometimes is unknown.

5.3 Blocked USB port

If an SRF stick is unplugged while it is in use, the USB port will block and will no longer be accessible. When the SRF stick is plugged back in, it will then have a different port number (port 1 instead of 0 for example). The system can be configured to use this port number instead, but care must be taken to ensure the correct port number is assigned to each SRF stick. The only known way to unblock a USB port is to restart the machine.

5.4 Robot struggling to move or turn

If the robot appears to be struggling to move, and in particular to turn, this is likely due to the batteries having insufficient charge. This results in the robot's motors stalling, which creates an audible whining noise. If this is the case, charge the batteries as described in Section 3.2. To prevent this situation occurring during a match, always ensure the batteries are fully charged before the match.

A RF configuration guide

A.1 Configuring the SRF stick

- Plug the SRF stick into a USB port on the DICE machine. Note the port number for this connection (ports are numbered by the order of USB devices inserted, starting with 0).
- Run the following terminal command:

```
$ screen /dev/ttyACMX 115200     //X is the port number for the SRF stick
```

  This opens an application called screen, which can be used to send information to the SRF stick.
- Once in the screen application (a blank terminal window), enter command mode by typing either `+++` or ∼∼∼, (depending on the initial configuration of the SRF stick). Do not press enter.
- In order to send configuration commands to the SRF stick, the following points should be noted:
  - Command mode automatically exits after five seconds of no commands.
  - The default settings for screen mean you cannot see what commands you type.
  - Press enter after each command.
  - Command sequences may have to be entered in one session, without exiting command mode.
  - To confirm that a command has been accepted, press `ctrl-a shift-c` to clear the screen terminal output after every command. This way, an `OK` means the command has been accepted, `ERR` means the command has been entered incorrectly and if nothing is displayed, command mode has been exited (due to the five-second timeout).
- Screen should display `OK` once command mode has been entered. Execute the following commands to reset the SRF stick to factory settings:

```
ATRE
ATAC
ATWR
ATDN
```

- Now enter command mode again (it will be `+++` this time). Configure the SRF stick by entering the following commands:

```
ATID00XX //XX is the group number (09)
ATAC
ATRP1    //to enable remote programming
ATAC     //to enable remote programming
ATCNXX   //XX is the hexadecimal number for the group's frequency (50)
ATAC
ATWR
ATDN
```

- Screen can be exited by typing `ctrl-a k`.

A.2 Configuring the Arduino

The Arduino is configured in the same way as the SRF stick, with the following changes to the instructions in Section A.1:

- Connect the DICE machine to the Arduino via the USB to Micro USB cable, and enter screen with the following command:

```
$ screen /dev/ttyACMX 115200    //X is the port number for the Arduino
```

- After resetting to factory default settings, add the following two lines before `ATWR` in the configuration command sequence:

  ```
  ATBD1C200
  ATAC
  ```

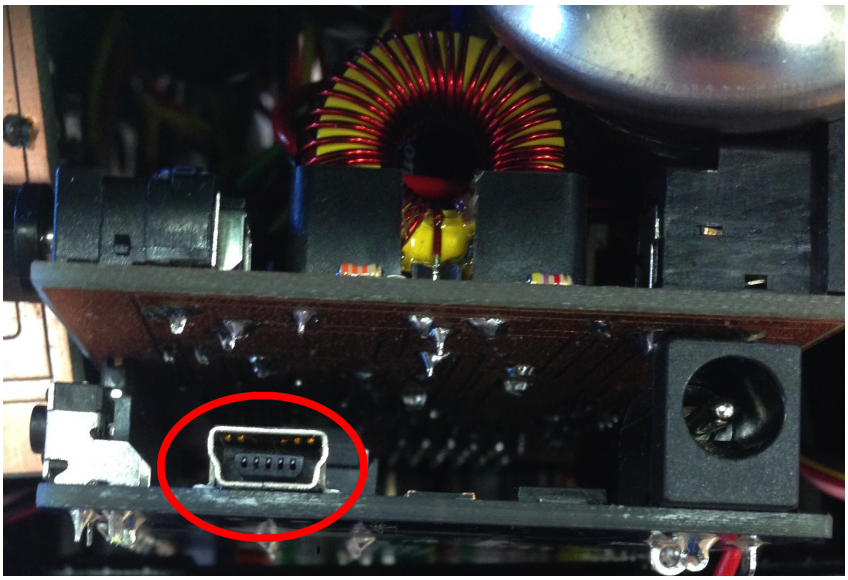

Figure 9: The Micro USB port on the Arduino

B Arduino programming guide

B.1 Launching the Arduino IDE

Launch the Arduino IDE on DICE by opening a terminal and typing the following command:

```
$ arduino
```

Setup the IDE to work with this project's sketchbook (folder for storing libraries) by going to `Arduino->Preferences` and changing 'Sketchbook location' to `[project-root-directory]/arduino/sketchbook`. This need only be done once per DICE account.

B.2 Programming the Arduino

- Launch the Arduino IDE, as described above.
- Connect the DICE machine to the Arduino with the USB to Micro USB cable.
- If the code involves using any IO device or motor other than Serial communication, make sure the battery pack is plugged into the Arduino.
- Load or write the desired code. This document assumes the robot is programmed with the Arduino sketch located in: `[project-root-directory]/arduino/final/`
- Click the upload button on the Arduino IDE.

If you are having problems, seeing the full error message may help solve the issue. To see the full error message, go to `Arduino->Preferences` and enable 'Show verbose output during... compilation and upload'.