User Guide

SDP Group 8-D

Karolis Greblikas
Tom Macmichael
Sanyam Yadav
Valentas Dicevicius
Keith Donaldson

The University of Edinburgh

Contents

1 Introduction

This document will guide you through the process of using the robot designed by Group 8 for the 2016 System Design Project.

2 Downloading and Setup

All of the source code is stored on GitHub. You can download the code to a DiCE machine by issuing the following command in the terminal:

```
git clone https://github.com/maccery/sdp-group-8.git
```

The source files will then be stored in the local directory named `sdp-group-8`, which you can access by giving the terminal command:

```
cd sdp-group-8
```

There is one dependency that needs to be installed: pySerial. This can be done on a DiCE machine by issuing the following command in the terminal:

```
pip install pyserial --user
```

Connection to the Arduino will not work without pySerial. You should now have all of the source code on your computer and pySerial installed.

3 Programming the Robot

There is one file that is used to program the robot, and this file has to be uploaded onto the robot via a USB-to-MicroUSB cable. Follow these steps:

- Open up the application Arduino, by typing `arduino` on the DiCE terminal.
- Load the `controller.ino` file from `arduino/controller` in Ardunio from File > Open.
- Connect the robot to the computer with the micro USB cable.
- Unplug the battery from the robot.
- Verify the code by clicking the 'tick' icon.
- Upload the file onto the Arduino by clicking the 'up arrow' icon.
- Unplug the USB cable and reconnect the battery on the robot.

The Arduino will now have required program installed onto it such that we can remotely control it from a computer.

4 Connecting to the Robot

The robot connects to the computer via a Ciseco SRF stick using the 0x47 (hex) frequency setting for ATCN. Please read the SDP Equipment Guide[1] for more information on the SRF stick settings. To connect to the robot, follow these steps:

- Make sure the battery is charged and plugged in to the robot.
- Make sure the micro USB cable is unplugged from the robot.
- Insert the SRF stick into the control PC.
- Launch Terminal.
- Navigate to the `sdp-group-8` folder.

The connection happens automatically when the control code is launched. For manual control – that is, giving the robot commands such as `kick` one-by-one – read §5.1. To use the vision system and have the robot be controlled autonomously, read §5.2.

5 Controlling the Robot

5.1 Manual Control

Manual control of the robot uses the `controller.py` file located in the `communication` directory. All of the commands are given as method calls in the Python interpreter. There are two lines required to set up the communication:

```
[DiCEPC] python
>>> from communication import Controller as ct
>>> tmp = ct("/dev/ttyACM0", ack_tries=10)
```

Note, `"/dev/ttyACM0"` may have to change to `"/dev/ttyACMx"` where x is the port number that the RF stick is on. However, 0 is the most likely and 1 is the likeliest alternative. The controller object is now held in `ct`, and you can issue commands as normal methods to the object `ct`.

Basic Functional Commands

The `move_duration` method takes one parameter: the duration (`float`) for which the robot will move in milliseconds. A negative value for duration will make the robot move backwards for that duration. An example usage is:

```
>>> ct.move_duration(1000)
```

The `turn` method turns the robot clockwise and takes one parameter: the duration (`float`) for which the robot will turn in milliseconds. A negative duration value will turn the robot anti-clockwise for that duration. An example usage is:

```
>>> ct.turn(100)
```

[1] http://www.inf.ed.ac.uk/teaching/courses/sdp/equipmentGuide.pdf

The `kick` method kicks the ball and takes one parameter: the power (`float`) for which the robot will kick the ball. This value x must be -1.0 << x << 1.0. A negative value will turn the kicker backwards at a given value. Note that you should not need to turn the kicker backwards, as the method does this for every use of the `kick` method. An example usage is:

```
>>> ct.kick(0.5)
```

The `grab` method closes the grabbers and takes no parameters, and the `ungrab` method opens them. Both methods take no parameters, and the usage is:

```
>>> ct.grab()
>>> ct.ungrab()
```

More information on commands which have less to do with core movement can be found in the Technical Specification. However, these commands will get you started with manually controlling the robot.

6 Tasks

Overview

The robots strategy system is made up of tasks. These can be separated into several categories: strategies (for playing a full game of football), large tasks, and sub tasks. The system is designed so that all three can be controlled from the interface.
A strategy is made up of large tasks which are made up of sub tasks.

To start the task runner, simply run

```
python Runner.py
```

After starting, the program will first do automatic vision calibration - this will take several seconds. Shortly after this it will initiate a connection with the Arduino. Once both of these tasks have completed successfully you will be prompted to enter which side of the pitch you are on. This is defined as the side seen by the vision system feed and by the user. Hence, left hald side of the pitch is also left, therefore the user would type in 'left' and vice versa.

Defining Teams

To enable system to accuractely track who is who, it needs to know the colour schemes of the robot. In Lines 106-110 of Runner.py are the colour definitions. A teammate does not strictly have to have the same team colour on the robot. As long as there are no duplicates in (team colour, group colour) pairs then there will be no issue.

The (team, group colour) definitions help the robot successfully avoid collisions with other robots, help with passing the ball and the also system knows where the robot being controlled currently is.

Task runner

After connecting to Arduino and selecting sides, youll be prompted to enter a task. This can be any of the tasks below. The task will continue to execute until its completed. The system

uses an iterative approach with tasks and so will keep looping until it is happy that the task has been successfully completed; only then will you be able to send it another task.

6.1 Strategies

These are made up of smaller tasks and are designed to never complete. Once started it will continue forever until the robot is disconnected. They are fully autonomous with no input required for a game of football.

Attacker

In attacker mode the robot will strive at all points to go get the ball, grab it, and kick it in the other teams goal.

```
>>> task_attacker
```

Defender

In defender mode the robot will stay in the defending region at all times. Its task: if the ball is in the defending region, go and grab the ball and then pass it to the teammate. If the ball is in the attacking region, sit at the midpoint between its goal and the ball.

```
>>> task_defender
```

Kicking off

Kicking off is simple: simply place the robot in the centre of the pitch and run one of the following two commands. The robot will kick the ball into the other teams half (as defined by the rules) and then play a game of football as per usual.

```
>>> task_attacker_kick_off
>>> task_defender_kick_off
```

Taking penalties

These tasks can be used to take a penalty and defend the goal in a penalty situation.

```
>>> task_goalie
>>> task_penalty
```

6.2 Smaller tasks

The strategies are made up of these subtasks that can be performed.

```
task_vision
task_move_to_ball
task_kick_ball_in_goal
task_move_and_grab_ball
task_rotate_and_grab
```

7 Hardware

7.1 Design

The diagram shown in figure 9.1 shows the design of the robot and the major components, beside the actual underside of the robot as shown by figure 9.2. This diagram takes a view from the bottom of the robot, with the colours representing the following:

Black: Wheels
Red: Ball Bearings
Orange: NXT Motor
Blue: Technic Mini-Motor
Green: 16-tooth gear
Grey: Connecting rods

The robot grabs the balls with grabbers at the two front ends of the robot. The gears at the front turn a set of "boomerang" shaped lego pieces and force the ball into the crevasse in the 'U' shape. The grabbers then release the ball and the kicker kicks the ball.

The robot turns by forcing one wheel forward and one wheel backwards concurrently. The ball bearings move freely with this allowing a very small rotation circle. The robot is fairly quick and, although it doesn't strafe, can turn sharply.

7.2 Maintenance

The battery pack is easily accessible. You can detach the two beams on the right edge of the robot beside the battery pack to slide it out the side, or simply remove it from the top. Make sure the cable attaching it to the Arduino is unplugged.

The Arduino reset button can be found on the left edge of the Arduino. To find the micro-USB port, gently life the Arduino straight up and it should detach from the mount. The port is on the bottom of the Arduino. Once completed, line up the pegs on the mounting mechanism with the wholes in the mount, and push down firmly to reconnect.

8 Troubleshooting

8.1 Connection Problems

If the computer is not receiving the "Ready" signal from the robot, there are a couple of obvious steps to rectify the issue:

- Make sure the SRF Stick is plugged into the computer.
- Close the program and retry.
- Restart the Arduino by disconnecting the battery and then reconnecting.
- Change or reboot PC.
- Reupload the controller code onto the Arduino.

If the robot doesn't seem to be doing what it should, this is most likely because it is losing packets. Try the above steps before trying the following:

- Turn off phones in the vicinity.
- Make sure the robot isn't more than 5m from the controlling computer.
- Unplug any other SRF sticks in the vicinity.

8.2 Robot isn't moving

When using our task runner you will see output in the terminal screen about whether a task has completed or not. Seemingly unexpected behaviour can often be explained by the systems collision system known as safety check. If a task would result in a collision into either the wall or another robot then this is known as a safety check breach and this will be printed into the terminal. In these situations the ball is likely to be in a position too close to the wall - once the ball is moved the robot will continue with its task without any intervention.

8.3 Robot is moving in wrong direction

- Ensure that the team, group colours are correctly defined in the code
- Ensure that there are no overlapping team, group colours in the code
- Ensure that every robot on the pitch has the correct colours and that there isnt any duplicates
- Ensure that the correct side of the pitch is entered when starting, as seen by the vision
- Restart the task runner

8.4 Code Problems

If you are receiving dependency issues, make sure that you have installed pySerial as outlined in §2. If that doesn't work, make sure that you have cloned the latest version from master from https://github.com/maccery/sdp-group-8.git.

If you are receiving generic Python errors, then there could be a bug in the code, however the more likely scenario is a corrupted file name, directory etc. In this case, clone a clean version of the working repository to your computer and use that code.

9 Appendix

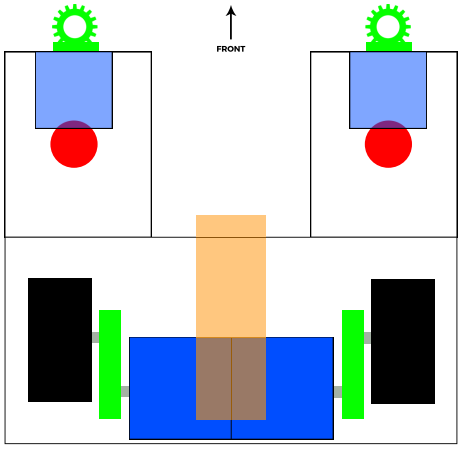

Figure 9.1: Diagram of Robot's Main Components

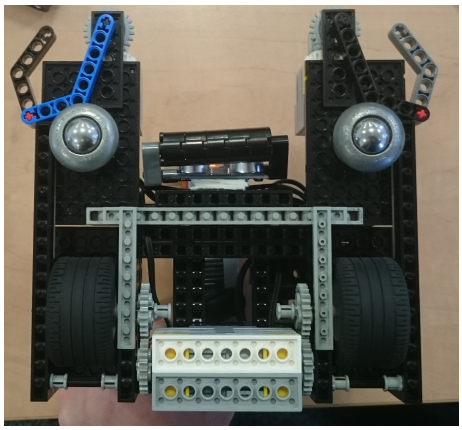

Figure 9.2: Bottom-view of the Robot