SDP User Guide

Group 11

Thomas Kerber, Marek Strelec, Euan Hunter

April 19, 2016

1 Setup

1.1 Requirements

In order to run the computer side control program, the following python packages need to be installed:

- `Polygon2`, for handling interactions of 2D polygons.
- `pyserial`, for interacting with the serial RF device.
- `numpy`, for various numerical calculations.
- `openCV`, for vision processing.

On the university DICE machines, `numpy` and `openCV` are already preinstalled. To install the remaining packages, run the following commands:

```
$ pip install --user Polygon2==2.0.6
$ pip install --user pyserial==2.7
```

1.2 Arduino

The Arduino must be programmed with the robot's firmware before use. If this is not already the case, it can be programmed by running the `arduino` program, and opening the `/arduino/group11/group11.ino` sketch. If not the case already, the sketchbook location should be set to `/arduino/`, to ensure that the libraries used are found. Next, the Arduino should be connected to the computer, and the sketch compiled and uploaded to it using the button for this purpose. The USB cable should then be removed. Finally, if the robot was on during this process, it should be briefly switched off to allow the Arduino serial device to reset.

2 Robot Structure

The robot has four sideways-facing holonomic wheels, one at each corner, which can move the robot sideways, and spin it on the spot. It can not move forwards, aside from turning and moving sideways. At the base of the robot, on the front side, a solenoid kicker is mounted. This is extended to kick the ball, and reset when the ball is grabbed. For this purpose, two grabber arms, which rest on the left and right sides of the robot, swing around the front-left and front-right corners to close around the kicker, pushing the ball into it. At the rear of the robot, a battery pack holding four lithium-ion batteries is mounted. This is secured by diagonal LEGO bars, which must be removed to access the battery pack.

3 Usage

Before use, ensure that the four lithium-ion batteries are fully charged. The charger is shared with Group 12. Ensure that the batteries are properly inserted into the two battery holders at the rear of the robot. To turn on the robot, flick the switch at the base of the battery packs. A red light will be visible from the Arduino, indicating that it is powered.

The robot is operated together with Group 12's, it is assumed that their robot has also been switched on at this point. To begin with, the device paths of both groups USB RF sticks must be determined. To determine these, first insert the RF stick for Group 11. Then run '`ls /dev/ttyACM*`' to list all serial USB devices available. This should return only one path, which can be noted as the path for Group 11's RF stick. Group 12's device path can then be determined by inserting their RF stick and re-running '`ls /dev/ttyACM*`'. One new path should be among those returned, which can be noted as the path to Group 12's RF stick.

Further information required to run the control unit is: The room being played in (3.D03 or 3.D04), which side of the pitch, left or right, is ours (viewed from the door), and which colour our team will have (blue or yellow). Armed with this knowledge, make sure that a top plate of each colour, as well as the ball, are on the pitch. Then, `cd` into the project root directory, and execute the following command:

```
$ ./main -p game -1 <PATH1> -2 <PATH2> -c <COLOUR> --pitch <NUM> -g <SIDE>
```

Using the following values for the arguments:

- The path of Group 11's RF stick for `<PATH1>`.
- The path of Group 12's RF stick for `<PATH2>`.
- '`y`' or '`b`' for `<COLOUR>`, if our team is yellow or blue respectively.
- '`0`' or '`1`' for `<NUM>`, if the game is being played in room 3.D03 or 3.D04 respectively.
- '`left`' or '`right`' for `<SIDE>`, if our goal is on the left or the right side of the pitch respectively.

A few windows should open up, the most prominent one of which displays a still of the pitch for calibration.

3.1 Calibration

The still of the pitch should show all four top plates and the ball clearly, as positioned earlier. The terminal will prompt the operator to click on samples of red, blue, yellow, green, and pink in turn, for the ball and the top-plate parts of the respective colours. This calibrates for the colours, ensuring that they are accurately detected. Of each colour, multiple samples should be clicked, when possible from different locations on the pitch (e.g. the green circles of all top plates, and not just one should be sampled). After several samples of each colour have been clicked, press the 'q' key to move on to the next. The colours are queried in the order listed above, and each colour is prompted by the terminal. Once calibrations are complete, they are saved locally. Due to changing light conditions, recalibration is necessary every few hours, however it is possible to skip the calibration phase and use the saved values by pressing the escape key at the start of calibration.

3.2 Online Usage

After the vision is launched, a window named 'Filter' will visualise the objects recognised from the vision feed, with objects drawn on it representing the robots and the ball (if they are found). Robots will be represented by an inner circle for the team colour (yellow or blue), an outer circle identifying which of the two team mates it is (pink or green, referring to the colour of the majority of circles on the top plate) and an arrow giving the orientation of the robot. The red ball is simply shown by drawing a red circle around it. Further, two other windows will be open, containing several trackbars. These trackbars are for filters that you can add to the output feed. There are filters to show only specific colours, or to different effects to the frame. For day to day usage, these do not need to be used.

While the control program is running, the overall strategy and logging can be modified. Entering '`debug`', '`info`', '`warn`', or '`error`' into the terminal, and pressing enter, sets the logging level appropriately. Further, a window to select the game state will have opened, allowing selection of the game state from '`kickoff-them`', '`kickoff-us`', '`penalty-them`', '`penalty-us`', '`normal-play`', and '`stop`'. Pressing any of these sets the game to the respective state. At startup, the state is '`stop`', and our team's robots will remain idle. Once the referee orders a kickoff, either '`kickoff-them`', or '`kickoff-us`' should be pressed, depending on if the opposing team, or our team has kickoff respectively. Likewise, should the referee call for a penalty kick, either '`penalty-them`' or '`penalty-us`' should be pressed, depending on if the opposing team or our team take the penalty respectively. The '`stop`' button can be pressed whenever the referee stops the game. Although it should not have to be used typically, pressing '`normal-play`' skips the kickoff phase of play, and immediately begins playing. This may be useful if the program had to be restarted mid-match. The game state will transition to '`normal-play`'

naturally from a kickoff, once the ball starts moving.

At halftime, both teams switch sides. The control program needs to be informed of the switch, either by typing '`switch`' into the console, and pressing enter, or by closing the control program and re-running it with the other pitch side. To terminate the control program, press 'Ctrl-D' with the terminal window selected. If this fails, press enter and try again.

4 Troubleshooting

Below are the actions to take for various common problems which may occur. The solutions are not guaranteed to work, but have in experience been effective.

4.1 The Robot Doesn't Move

If the robot is not moving, the first thing to do is to check the batteries. Ensure that they are present, and charged. Further, confirm that the robot was switched on. If the Arduino has a red light on, it is receiving power and the issue is more likely to be one of communications. In this case switch the robot off for a few seconds and turn it on again to reset the serial device.

4.2 The Control Program Doesn't Start

The control program not starting is likely due to the RF control signals blocking. To test this, run the control program without the argument for the RF stick. If it runs, then try waiting for five seconds, then re-running with the normal arguments. If after a few attempts, this also does not work, then double check the guard characters set on the RF stick. They should be three tilde characters ('`~~~`'). The proprietary CISCO tools located in the '`/tools`' folder can be used to check this, and, if needed, to reset them.

4.3 The Vision is Misdetecting Objects

The vision misdetecting objects is likely due to miscalibration. Close the controller, restart and recalibrate. Take care to select a wide range of colour samples, including the fringes of the circles to detect.

4.4 The Robot is Very Slow

The batteries are low. Recharge or replace them.

4.5 The Vision is Solid Green

The video input is set to draw from the wrong source. Close the control program, and run '`xawtv`'. In the options, select svideo as the source.

4.6 The Kicker Doesn't Work

Double check the I/O port connections with the kicker. Try also listening at the breadboard during a kick for an audible click. If this can be heard, the issue is in the wiring from the batteries to the breadboard, or from this to the kicker. If not, the issue is either the connectors being plugged into the wrong I/O port (it should be the furthest inward on the I/O board), or a faulty connection from the I/O port to the breadboard.