User Guide

SDP Group 15-H

April 16, 2016

1 Installation

1.1 Cloning the repository and setting up

To clone the repository, execute in terminal:

```
$ git clone https://github.com/julijonas/venus.git
```

Then go to the root directory of the project and create a Python virtual environment which will contain a local copy of the libraries needed for this project:

```
$ cd venus
$ virtualenv --system-site-packages env
$ source env/bin/activate
```

Install the *pyserial* library which is used to send data to Arduino through RF stick:

```
$ pip install pyserial
```

To set the persistent radio chip configuration for a new Arduino board, connect to the PC with an USB cable, execute `screen /dev/tty0` in terminal, and enter the following commands substituting the appropriate group number and radio frequency band:

```
+++
ATID0015
ATAC
ATRP1
ATAC
ATCN80
ATAC
ATWR
ARDN
```

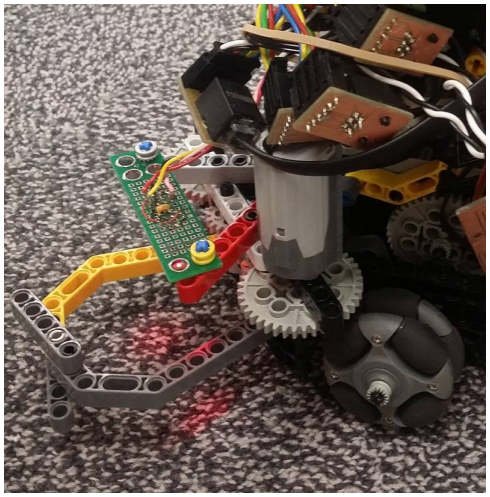

Figure 1: The grabber

To close `screen`, press *Ctrl+A* and then *X*. The identical steps have to be performed for a new RF stick.

To program the Arduino to receive and execute messages, open the Arduino IDE by executing `arduino` in terminal. You'll need to add three libraries which can be found under `arduino/` in the project directory: *ArduinoSerialCommand*, *SDPArduino*, *SimpleTimer*. In order to add a library, go to `Sketch -> Import Library... -> Add Library...` and choose the library directory you want to import. Then open the Arduino file `arduino/arduino.ino` using `File -> Open...` and upload it to the board using `File -> Upload`.

2 Hardware overview

2.1 Components

2.1.1 Wheels

The robot has four holonomic wheels, which are placed in the middle of each side of the robot. The holonomic motion allows the robot to move in any direction. All wheels are powered by NXT motors.

2.1.2 Grabber and Kicker

The grabber consists of two symmetric parts resembling arms (Figure 1). It is powered by a single Power Functions Medium motor, which is connected to a symmetric gear system. The latter is used to ensure both arms are opened and closed at the same time. The robot does not have a kicker. Instead it opens the grabber while turning to make a kick. In doing this the centrifugal force holds the ball towards the end of the grabber and the ball is in contact with the claw that will apply the kicking force.

2.1.3 Rotary encoder board

Each NXT motor is connected to the rotary encoder board (Figure 2). The connections in the anticlockwise order from the top are: the I2C bus to the Arduino, back right motor, back left motor, front left motor, and front right motor. Using this board the information about the amount of rotations the motor has performed since the last query is available for the Arduino code as a separate integer for every motor. Every 5 ms the board is queried whether the target value has been reached. After the average of the rotary values of all four motors becomes greater or equal to the target value, the motors are stopped.

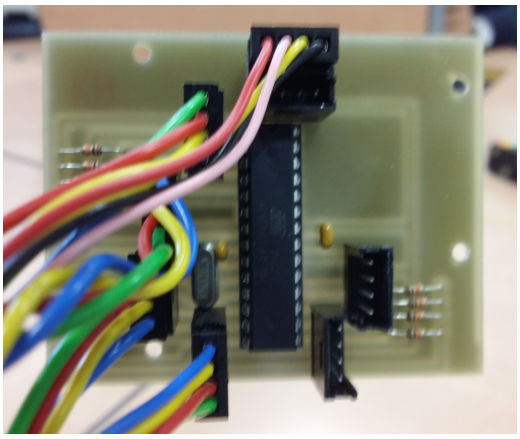

Figure 2: The rotary encoder board

2.1.4 Light sensor

The light sensor is located above the grabber (Figure 3). It is used to check whether the robot has successfully acquired the ball after grabbing. The sensor provides an integer value that increases the greater the distance to the nearest object in its direct line of sight is. Then the Arduino compares the integer to a predefined threshold corresponding to the red ball. Sometimes a white line inside the pitch can be mistaken for the ball due to their similar sensor values.

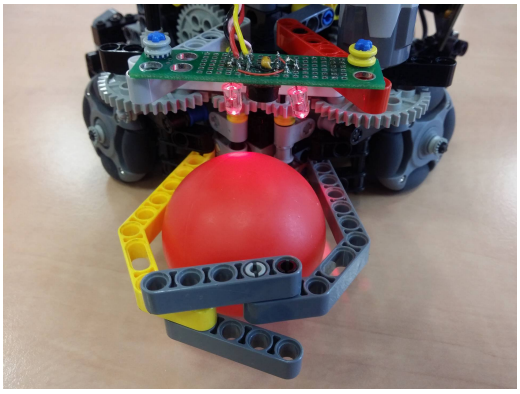

Figure 3: The light sensor with the ball grabbed

2.2 Usage

2.2.1 Preparation for the match

In order to turn the robot on, connect the battery pack to the power board. The pack consists of 10 AA rechargable batteries. It is advisable to swap batteries after each match, even if the batteries are not fully discharged at that point. Normally one fully charged battery pack would last for 7-10 minutes. After that the performance of the robot might decrease depending on the amount of actions it made.

In order to swap the batteries, take them out, put them to charge and insert the new batteries into the battery pack. Easy!

2.2.2 RF stick

To communicate with Venus the RF stick should be connected to the machine you are running the commands from. Then after performing steps described in "Running instructions" section, you will be able to operate the robot.

If you make any changes in the Arduino code, you need to upload them before running the commands (uploading instructions can be found in the "Installation" section above). Uploading can be done either via RF stick or the USB cable connected to the Arduino board.

3 Software overview

3.1 Running instructions

Run the following command in the terminal from the project root directory if it has not been done already to enable the Python virtual environment which contains the required libraries:

```
$ source env/bin/activate
```

In order to set the room containing the pitch, colors of the robot, and side of the goal, change the hard-coded parameters of *init* method in *control/holonomic.py* as detailed in Table 1. For example, if the room is 3.D04, the robot has the yellow team color and one green-coloured dot, and the robot is defending the goal closer to the computers, the line would be:

```
def init(self, room_num=1, team_color='yellow', our_color='green',
    computer_goal=True):
```

Then ensure that the RF stick is plugged in. To start the application, make connections to the RF stick, vision feed, and get access to the command prompt, execute the following line:

```
$ python main.py
```

Room	room_num
3.D03	0
3.D04	1

team_color
'yellow'
'blue'

our_color
'green'
'pink'

computer_goal
True
False

Table 1: Allowed parameters for the *init* method

NB: In case you make changes to the Arduino code, you should upload your changes to the board as detailed in Section 1.1.

Then the operator of the robot can start entering the commands. The main command is `hs` which starts the strategy. The listing of all commands is provided in Table 2.

Constantly run the strategy state machine	`hs`
Output a picture of the potential field of the state	`map state_name`
Perform a holonomic motion (angles in degrees)	`move direction_angle turn_angle`
Move forward (in cm, negative means backward)	`f distance`
Rotate clockwise (in degrees, negative means anti-clockwise)	`c angle`
Stop all motors	`s`
Open the grabber	`o`
Close the grabber	`g`
Perform a spin kick	`ee`
Print world state	`w`
Query light sensor	`query_ball`
Pass the ball to the teammate	`pass_ball`
Catch the ball coming from the teammate	`catch_ball`
Kick the ball to the goal	`goal`
Exit the application	`exit`

Table 2: Commands available for the operator of the robot

To catch the ball from the teammate, the robot turns towards the teammate and opens its grabber. When the ball is kicked towards the robot, the ball is grabbed. In case the ball bounces off, the robot starts moving to grab the ball. To pass the ball to the teammate, the robot turns towards the teammate and kicks the ball. The kick distance of the ball is calculated so that the ball can successfully reach the teammate.

3.2 Calibration of the vision

copy over from spec

3.3 Handling

Handle the robot with care. It is advisable not to lose the top plate during the game. It can be secured properly to the robot using Blu-Tack.

3.3.1 Calibration

Before playing a match, ensure that the spin-kick (`goal` command), going forward (`f` command), and turning (`c` command) are calibrated. It can be done by repeatedly executing the respective commands with different input values, changing the data points in the `calibration.ods` spreadsheet, and changing the equations in `ee`, `f`, and `c` methods in `control/holonomic.py` to respective new computed equations from the spreadsheet.

When calibrating the spin-kick the aim is for the robot to be not fully turned towards the goal before the kick. To achieve that, there are different "correction angles" for six main zones of the pitch (Figure FIGURE). These angles can be calibrated in `strategy/simple_holonomic.py` in `shot_correction` method. The best strategy here would be to place robot with the ball in different sections of one zone and execute the `goal` command choosing the constant for the angle that achieved the best result.

Another important action is checking whether the light sensor threshold correctly identifies the grabbed ball and lack thereof. The threshold is defined in `query_ball` method in `control/holonomic.py`.

NB: If another robot is doing the kick-off in a game, do not start the strategy by typing `hs` before that robot performs the kick-off.

4 Troubleshooting guide

Problem 1: When uploading the code to the Arduino board, the following error message appears:

```
processing.app.SerialNotFoundException: Serial port '<port>' not found. Did
    you select the right one from the Tools > Serial Port menu?
```

Solution: Make sure you are not running the Python application at the same time or have not opened the serial interface any other way when the changes are being uploaded. The Arduino IDE is only able to program the Arduino when the serial interface of the RF stick or Arduino itself is registered as `/dev/ttyACM0` on the PC. If there are other programs accessing `/dev/ttyACM0` and the RF stick or USB cable is unplugged and plugged again, the "new" device is issued `/dev/ttyACM1` instead by the Linux kernel.

Problem 2: When sending commands the robot does not respond to any commands although it should move.

Solution: Power cycle the Arduino board.

Problem 3: The distortion and color ranges vary on different machines and pitches.

Solution: Recalibrate the vision system as detailed in Section 3.2.

Problem 4: The spin kick is not reliable.

Solution: Recalibrate the spin kick as detailed in Section 3.3.

Problem 5: The Python application cannot connect to the RF stick because the RF stick has registered itself as `/dev/ttyACM1`.

Solution: Either close all programs that have handles to `/dev/ttyACM0` and reconnect the RF stick or change the `device_no` parameter in the `connect` method in `control/holonomic.py` to 1 and restart the Python application.

Problem 6: The robot moves unusually slow.

Solution: Change the battery set to a fully charged one.

Problem 7: The commands are evidently sent but the robot does not move.

Solution: Power cycle the Arduino. If that does not help, change the battery set to a fully charged one.

Problem 8: The freshly charged batteries are hot.

Solution: Do not put such battery set into the Arduino as the motions will not be reliable. Wait for them to cool down and obtain another set instead.