

ROBOBULLS SOFTWARE PRESENTATION

A PROJECT DONE BY:



Shaughn Seepaul
seepauls@mail.usf.edu
Computer Engineering
University of South
Florida

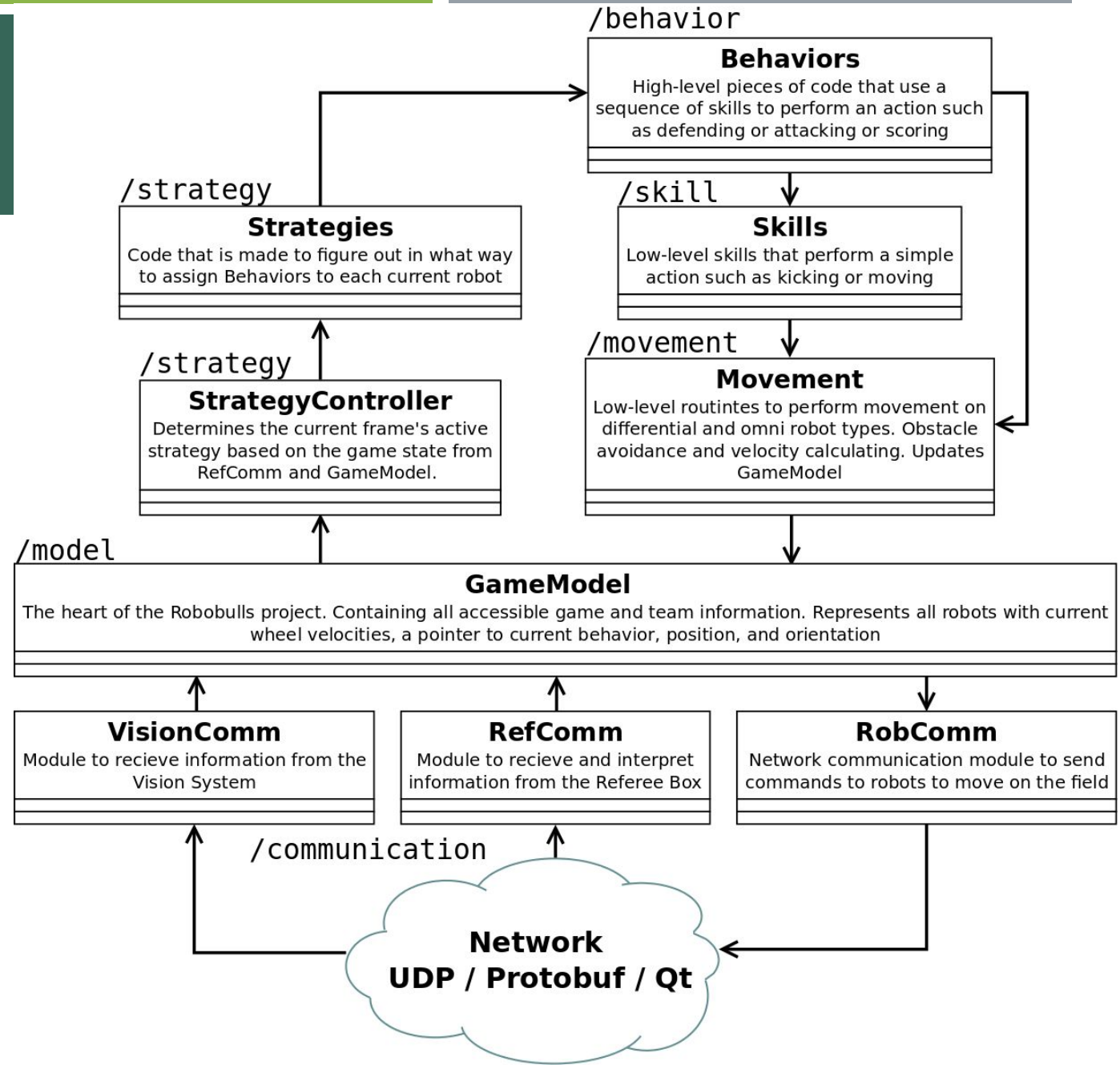
Abdul Munim Zahid
abdulmunimz@mail.usf.edu
Computer Science
University of South Florida

Syed Hasan
sthasan@mail.usf.edu
Computer Science
University of South
Florida

OVERVIEW

- ACKNOWLEDGEMENTS
- DOCUMENTATION
- SOFTWARE REQUIREMENTS
- INSTALLATION OF SOFTWARE
- COMPILING / RUNNING SOFTWARE
- ISSUES FACED
- DEMONSTRATION OF SIMULATION
- RESOURCES

RoboBulls Software Project Organization



ACKNOWLEDGMENTS

- We would like to thank Dr. Alfredo Weitzenfeld for his patience and resources for us to be able to complete the requirements of this project.
- We would also want to extend our gratitude to Muhaimen Shamsi for his guidance and taking the time to explain the previous software infrastructure.

DOCUMENTATION

The documentation for the updated software is available via <https://github.com/isMunim/RoboBulls> :

The screenshot shows the GitHub repository page for `isMunim / RoboBulls`. The repository has 830 commits, 10 branches, 0 releases, and 7 contributors. The latest commit is `d6451f8` from an hour ago. The repository is currently on the `master` branch. The file list shows various folders and files, including `arduino`, `behavior`, `communication`, `documents`, `examples`, `gui`, `hw/board`, `include`, `model`, `movement`, `robotc`, `skill`, `strategy`, `utilities`, and `xbeesniffer`. The repository is currently on the `master` branch.

isMunim / RoboBulls

Watch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights

No description, website, or topics provided.

830 commits 10 branches 0 releases 7 contributors

Branch: master New pull request

Create new file Upload files Find file Clone or download

isMunim Added makefile Latest commit d6451f8 an hour ago

arduino	Modified install script to always export serial rules	2 years ago
behavior	Updated install script to copy udev rules correctly. Refactored movem...	a year ago
communication	work in progress on motion stack	a year ago
documents	Merged changes in install_robobulls.sh. Kept inclusion of abosulte pa...	a year ago
examples	Added examples folder	3 years ago
gui	Fixed Obstacle avoidance crash for multiple robots by checking whethe...	a year ago
hw/board	Added power connectors to ctrl board, fix pwr board error	a year ago
include	GoToPosition now re-runs FPPA pathfinding for every frame.	a year ago
model	Merged changes to motion control and path planning	a year ago
movement	Merged changes to motion control and path planning	a year ago
robotc	On-field robot labels now correctly rotate as the camera rotates	4 years ago
skill	Updated install script to copy udev rules correctly. Refactored movem...	a year ago
strategy	Fixed Obstacle avoidance crash for multiple robots by checking whethe...	a year ago
utilities	Merged changes to motion control and path planning	a year ago
xbeesniffer	YisiBot Motion Working	2 years ago

SOFTWARE REQUIREMENTS

The Robobulls software requires 3 major pieces of software in order to run:

- grSim Simulation software
- The RoboCup-SSL Ref Box
- Most importantly, the AI code from GitHub

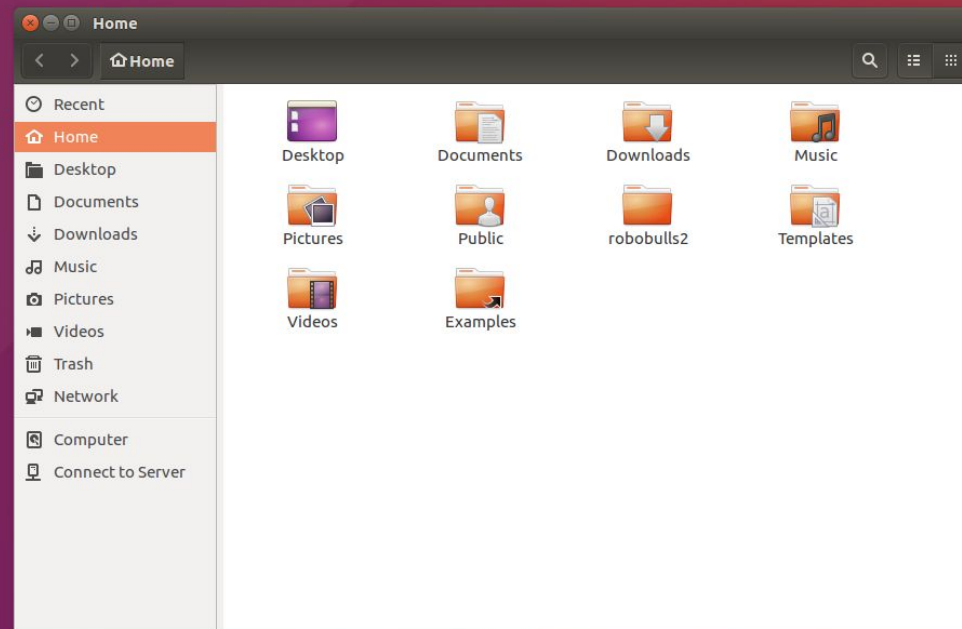
INSTALLATION OF SOFTWARE

Don't fear, installation is easy. By following the next few steps, you will have everything installed and ready to compile:

- In the command line of the linux terminal, install the git apt package by entering “sudo apt-get install git”.
- Afterwards, clone the github repository onto your computer by entering the command “git clone <https://github.com/isMunim/RoboBulls>”

INSTALLATION OF SOFTWARE

After completing these two tasks, the “robobulls” folder should show up on the home directory as shown below:

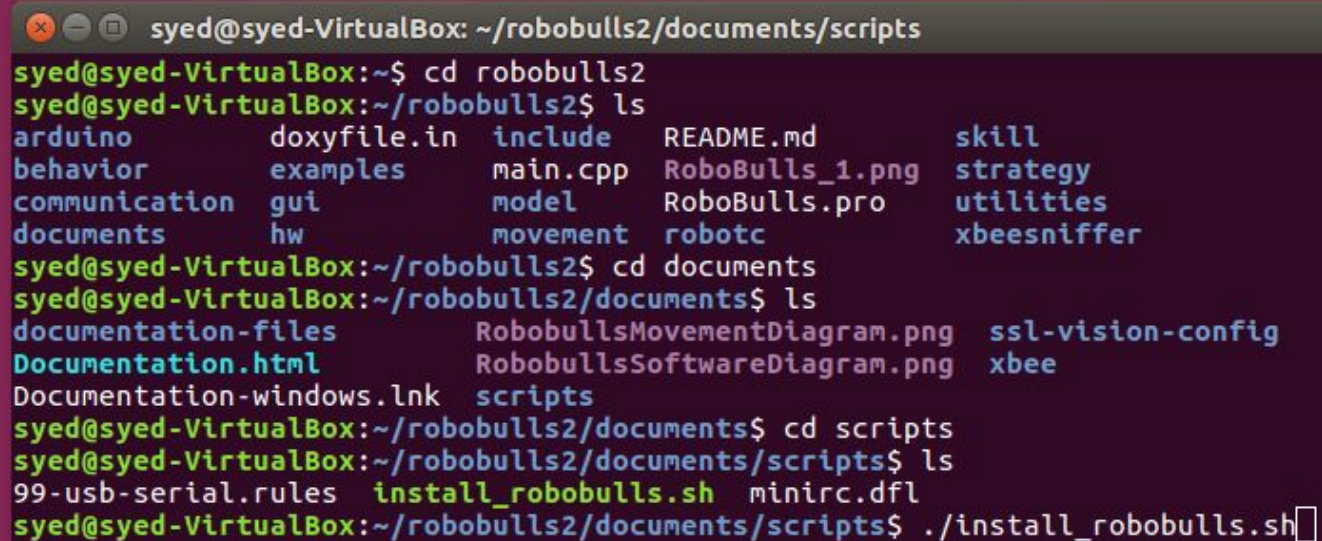


INSTALLATION OF SOFTWARE

- Next, navigate yourself in the terminal through the robobulls folder using the “cd <folder name>” command.
- If you are not familiar with the Linux/Ubuntu OS, you can enter the “ls” command. By doing this, a list of folders will be printed out.
- When using “cd” to go inside of the robobulls folder, you can then press “ls” again to see the available files/folders. Enter “cd documents” into the terminal. This changes the current directory to the “documents” directory in the robobulls folder.
- After this, go to the “scripts” folder by entering “cd scripts” and then press “ls” again.
- Run the installation script “install_robobulls.sh” using the command “./install_robobulls.sh”

INSTALLATION OF SOFTWARE

This script installs all the required software to run the Robobulls Software. A visual representation of is seen below:

A terminal window titled 'syed@syed-VirtualBox: ~/robobulls2/documents/scripts' displays the following commands and output:

```
syed@syed-VirtualBox:~$ cd robobulls2
syed@syed-VirtualBox:~/robobulls2$ ls
arduino      doxyfile.in  include      README.md    skill
behavior     examples     main.cpp     RoboBulls_1.png strategy
communication gui          model        RoboBulls.pro utilities
documents    hw          movement     robotc       xbeesniffer
syed@syed-VirtualBox:~/robobulls2$ cd documents
syed@syed-VirtualBox:~/robobulls2/documents$ ls
documentation-files  RobobullsMovementDiagram.png  ssl-vision-config
Documentation.html   RobobullsSoftwareDiagram.png  xbee
Documentation-windows.lnk scripts
syed@syed-VirtualBox:~/robobulls2/documents$ cd scripts
syed@syed-VirtualBox:~/robobulls2/documents/scripts$ ls
99-usb-serial.rules  install_robobulls.sh  minirc.dfl
syed@syed-VirtualBox:~/robobulls2/documents/scripts$ ./install_robobulls.sh
```

INSTALLATION OF SOFTWARE

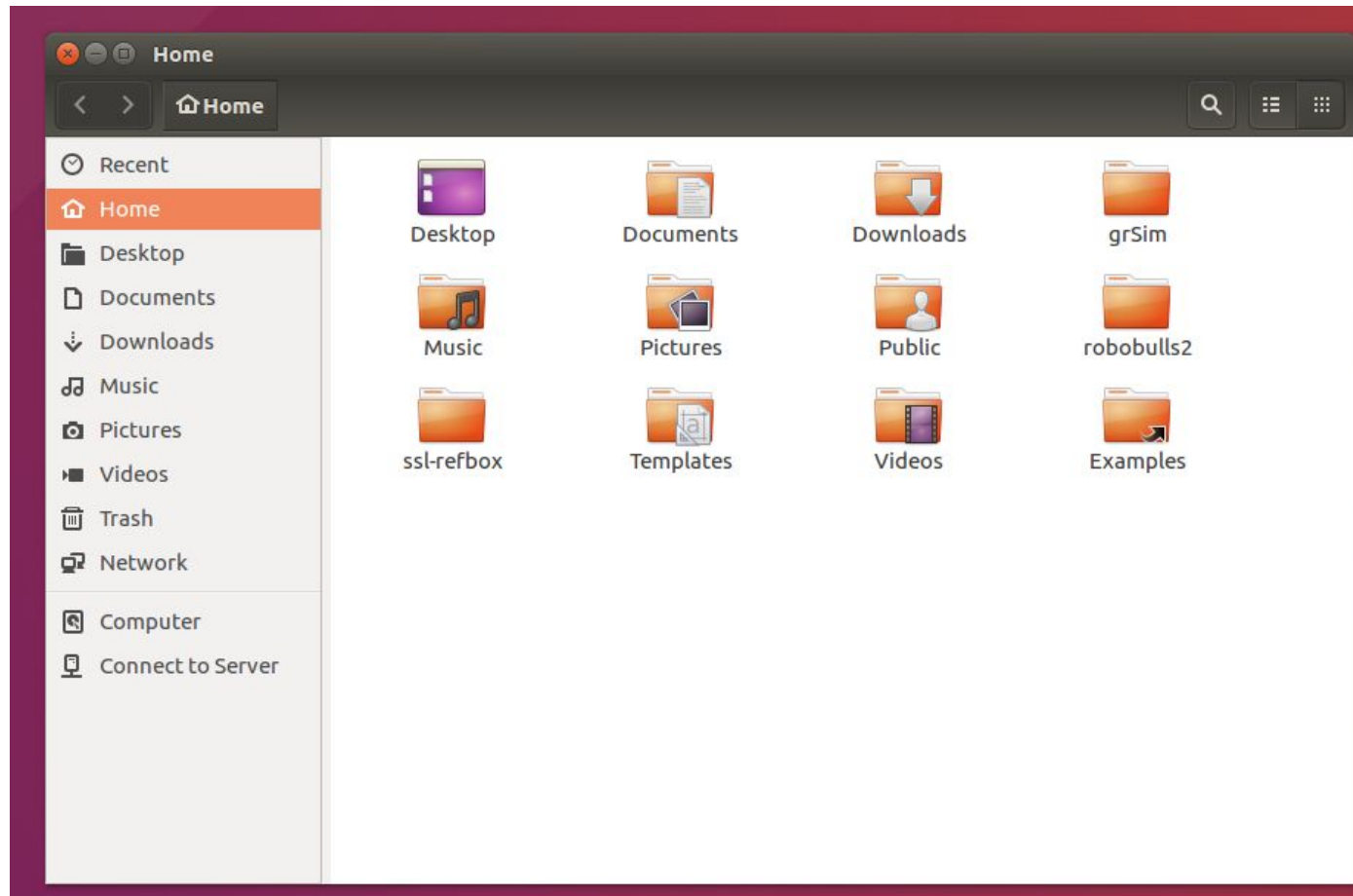
Below we have a visual of what happens after the script is run:

```
syed@syed-VirtualBox: ~/robobulls2/documents/scripts
Scanning dependencies of target client_automoc
[ 80%] Automatic moc for target client
Generating moc_mainwindow.cpp
[ 80%] Built target client_automoc
[ 82%] Running C++ protocol buffer compiler on ../../src/proto/grSim_Packet.proto
[ 84%] Running C++ protocol buffer compiler on ../../src/proto/grSim_Replacement.proto
[ 86%] Running C++ protocol buffer compiler on ../../src/proto/grSim_Commands.proto
Scanning dependencies of target client
[ 88%] Building CXX object clients/qt/CMakeFiles/client.dir/grSim_Replacement.pb.cc.o
[ 90%] Building CXX object clients/qt/CMakeFiles/client.dir/grSim_Commands.pb.cc.o
[ 92%] Building CXX object clients/qt/CMakeFiles/client.dir/grSim_Packet.pb.cc.o
[ 94%] Building CXX object clients/qt/CMakeFiles/client.dir/main.cpp.o
[ 96%] Building CXX object clients/qt/CMakeFiles/client.dir/mainwindow.cpp.o
[ 98%] Building CXX object clients/qt/CMakeFiles/client.dir/client_automoc.cpp.o
[100%] Linking CXX executable ../../bin/client
[100%] Built target client
RoboBulls2 already installed (directory exists)
Regenerating protobuf files
syed@syed-VirtualBox:~/robobulls2/documents/scripts$
```

```
syed@syed-VirtualBox: ~/robobulls2/documents/scripts
ll 3.5.1-1ubuntu3 [1,121 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 libjsoncpp1 amd64 1.7.2-1 [73.0 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 cmake amd64 3.5.1-1ubuntu3 [2,623 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 libpcrecpp0v5 amd64 2:8.38-3.1 [15.2 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu xenial/universe amd64 libqt5script5 amd64 5.5.1+dfsg-2build1 [707 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libudev-dev amd64 229-4ubuntu21.2 [150 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 qml-module-qtquick-controls amd64 5.5.1-1ubuntu1 [643 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 autotools-dev all 20150820.1 [39.8 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 po-debconf all 1.0.19 [234 kB]
Get:10 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 libfile-stripnondeterminism-perl all 0.015-1 [10.3 kB]
Get:11 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 dh-strip-nondeterminism all 0.015-1 [4,864 B]
Get:12 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 debhelper all 9.20160115ubuntu3 [739 kB]
7% [12 debhelper 0 B/739 kB 0%]
```

INSTALLATION OF SOFTWARE

After installation is complete, we can see the new folders appear among the others: (default installation path is Home)



COMPILING / RUNNING THE SOFTWARE

- Now that the installation is complete, we now compile the code by running the makefile that we created in the robobulls folder.
- To do this, we simply ensure that our current directory is set to the robobulls folder in the Linux terminal. If not, we execute the command “cd robobulls” after opening the terminal.
- After doing this, simply enter “make” into the terminal and observe the code compile.

COMPILING THE SOFTWARE - RUNNING THE MAKEFILE

```
Makefile — Edited v
#####
# Makefile for building: RoboBulls
# Project: RoboBulls.pro
# Authors: Abdul Munim Zahid(@isMunim), Syed Tahmid Hasan(@tahmid53), Shaughn Seepaul(@sdseepaul)
# Command: cd to root robobulls directory
# make
# Acknowledgements: gmake, qt creator, stackoverflow were used to create the makefile
#####

MAKEFILE      = Makefile

##### Compiler, tools and options

CC             = gcc
CXX            = g++
DEFINES        = -DQT_NO_DEBUG -DQT_NETWORK_LIB -DQT_WIDGETS_LIB -DQT_GUI_LIB -DQT_SERIALPORT_LIB -D
CFLAGS         = -m64 -pipe -O2 -Wall -W -D_REENTRANT -fPIC $(DEFINES)
CXXFLAGS       = -m64 -pipe -std=c++0x -O2 -Wall -W -D_REENTRANT -fPIC $(DEFINES)
INCPATH        = -I. -isystem /usr/include/x86_64-linux-gnu/qt5 -isystem /usr/include/x86_64-linux-g
qt5/QtWidgets -isystem /usr/include/x86_64-linux-gnu/qt5/QtGui -isystem /usr/include/x86_64-linux-g
gnu/qt5/QtCore -I. -I. -I/usr/lib/x86_64-linux-gnu/qt5/mkspecs/linux-g++-64
QMAKE         = /usr/lib/x86_64-linux-gnu/qt5/bin/qmake
DEL_FILE       = rm -f
CHK_DIR_EXISTS = test -d
MKDIR          = mkdir -p
COPY           = cp -f
COPY_FILE      = cp -f
COPY_DIR       = cp -f -R
INSTALL_FILE   = install -m 644 -p
INSTALL_PROGRAM = install -m 755 -p
INSTALL_DIR    = cp -f -R
DEL_FILE       = rm -f
SYMLINK        = ln -f -s
DEL_DIR        = rmdir
MOVE           = mv -f
TAB            =
```

```
ismunim@Munim-Surface-Pro-3: ~/robobulls2
ismunim@Munim-Surface-Pro-3:~$ cd '/home/ismunim/robobulls2'
ismunim@Munim-Surface-Pro-3:~/robobulls2$ make
```

RUNNING THE SOFTWARE

- When the compilation is complete, there will now be a grSim file generated called “RoboBulls”.
- In order for this file to work properly, the grSim simulation software (which was installed earlier) needs to be open simultaneously with the “RoboBulls” file.
- To do this, navigate through the grSim folder screen in the Home (root) directory and select the “bin” folder then select the “grsim” executable file. You will notice all the robots on the field with a menu on the left hand side of the screen.
- Afterwards, open the Ref Box by going from Home to the “ssl-refbox” folder and then to the “sslrefbox” executable file.

Configuration for teams

- **SIMULATOR_ADDRESS**

The IP address of the computer running the simulator (SIMULATOR_ADDRESS_LOCAL for local)

- **SIMULATOR_PORT**

Set to 2011 as defined by grSim

- **VISION_ADDRESS**

Set to "Vision multicast address" shown on the left-hand side of the simulator if simulated else set to the address of the computer running SSL-Vision

- **VISION_PORT**

Set to 10020 if simulated or 1002 if using SSL-Vision

- **REFBOX_ADDRESS**

Set to "224.5.23.1" and make sure you open SSL's refbox.

- **REFBOX_PORT**

Set to 10001 for now. See last 5 slides for issues with this.

- **REFBOX_LISTEN_ENABLED**

Setting this to 0 will enable test strategy, setting to 1 will enable the refbox and our AI.

BRANCHES FOR EASE OF USE

In the GitHub, there are 4 branches available for your selection:

- team-blue-config
Configuration for a team running on the Biorobotics Lab.
- team-yellow-config
Configuration for a team running on a separate computer not connected to BioRobotics lab cluster/IP
- simulation-fixed
A branch which includes to previously missing function needed to run the teststrategy.cpp as specified by documentation [here](#).
- ssl-league -fixed
A branch which is configured to run using the SSL-Vision and the Yisi Robot. Better to use the submission from Bo and Donglin as that is the more tested version.

BRANCHES FOR EASE OF USE

Active branches

`ssl-league-fixed` Updated 13 minutes ago by isMunim

0 | 1

 Compare

`team-yellow-config` Updated 22 minutes ago by isMunim

0 | 1

 Compare

`simulation-fixed` Updated 38 minutes ago by isMunim

0 | 2

 Compare

`team-blue-config` Updated an hour ago by isMunim

1 | 2

 Compare

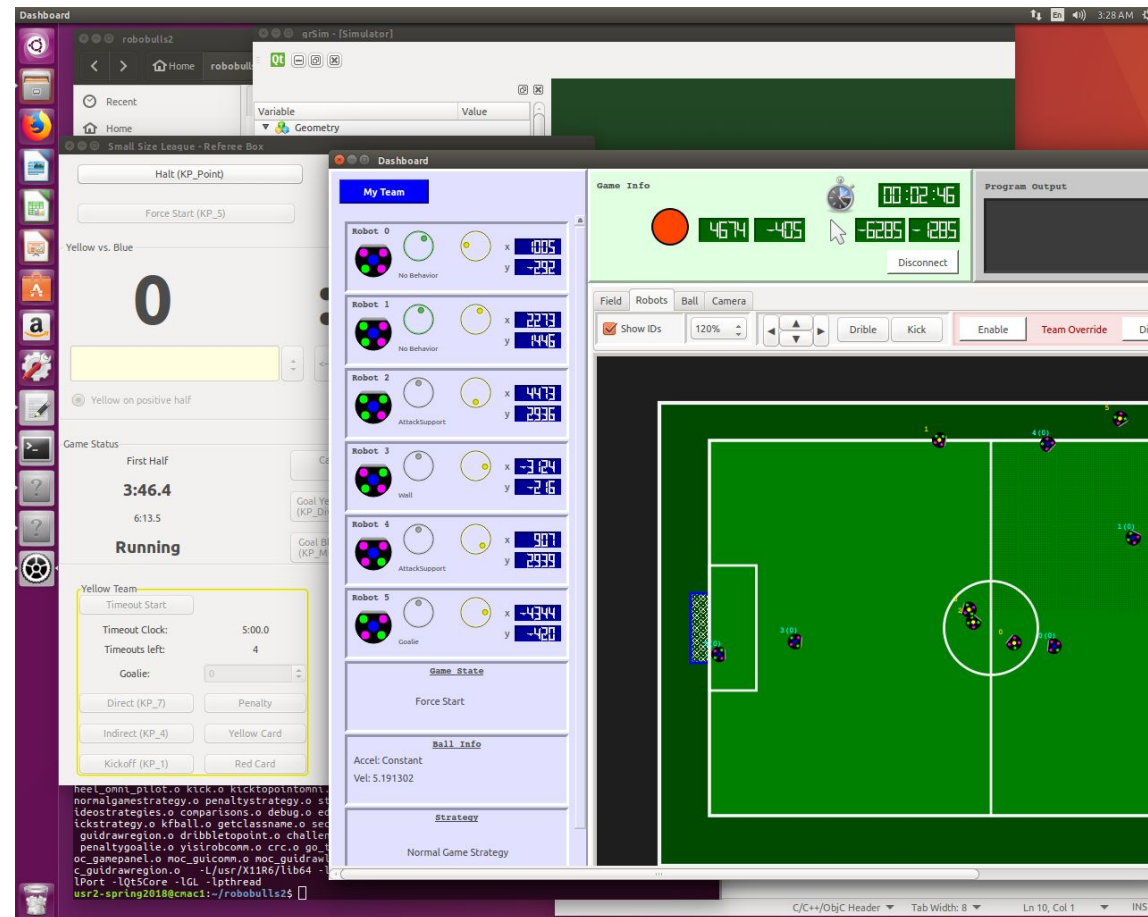
Running RoboBulls on the Simulator and performing a simple behavior

Here we are going to compile and run the project on the simulator and perform a simple **Behavior**. (see <..../robobulls2/documents/documentation-files/classBehavior.html>)

- Clone the simulation-fixed branch on your computer
- Open the RoboBulls.pro project in Qt Creator and compile. After running the installation script, there should be no compilation issues (Tested on Ubuntu 14.04 and 16.04 only with QT Creator 4.5.0 and QT 5.10.0)
 - Make sure you set the QT version in your QT creator by going into Tools->Options->Build & Run->Kits
- Open the grSim simulator (grSim/bin/grsim) from the script's install path on your computer or any other computer
- Go to [include/config/communication.h](#) and change **SIMULATOR_ADDRESS** to the IP (in quotes) of the computer running grSim (SIMULATOR_ADDRESS_LOCAL for local).
- Change **REFBOX_LISTEN_ENABLED** to 0–This will enable of the **TestStrategy** below.
- Change **VISION_ADDRESS** under the #if SIMULATED to the "Vision multicast address" shown on the left-hand side of the simulator
- Go to **strategy/teststrategy.cpp** and insert **gameModel->findMyTeam(0)->assignBeh<GoToBehavior>()**; under **TestStrategy::assignBeh**
 - You can also assign the following functions: KickBeh, DribbleToPointBeh, DribbleBackBeh, RotBeh, DribbleBeh, MotionTestBeh, Strafe
- Making sure **SIMULATED** in include/config/simulated is 1, Run the program.
- If these steps are completed correctly, **Robot 0** on TEAM should move to the center of the field.

COMPILING / RUNNING THE SOFTWARE

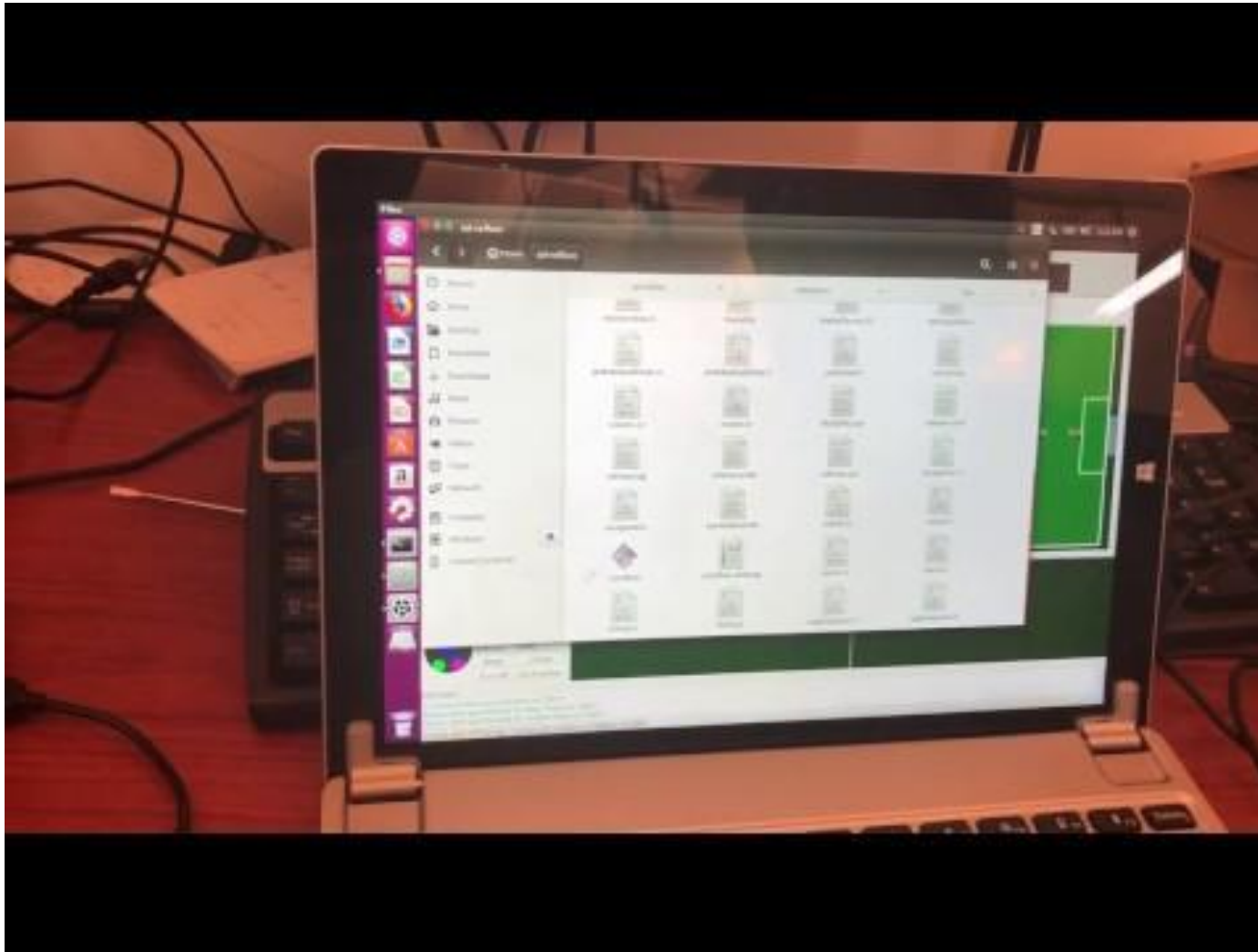
Here is an example of how the computer in the lab's screen should look like:



COMPILING / RUNNING THE SOFTWARE

- Once the simulation is up, just hit disable on both robobull instances and the refbox will take over from there.

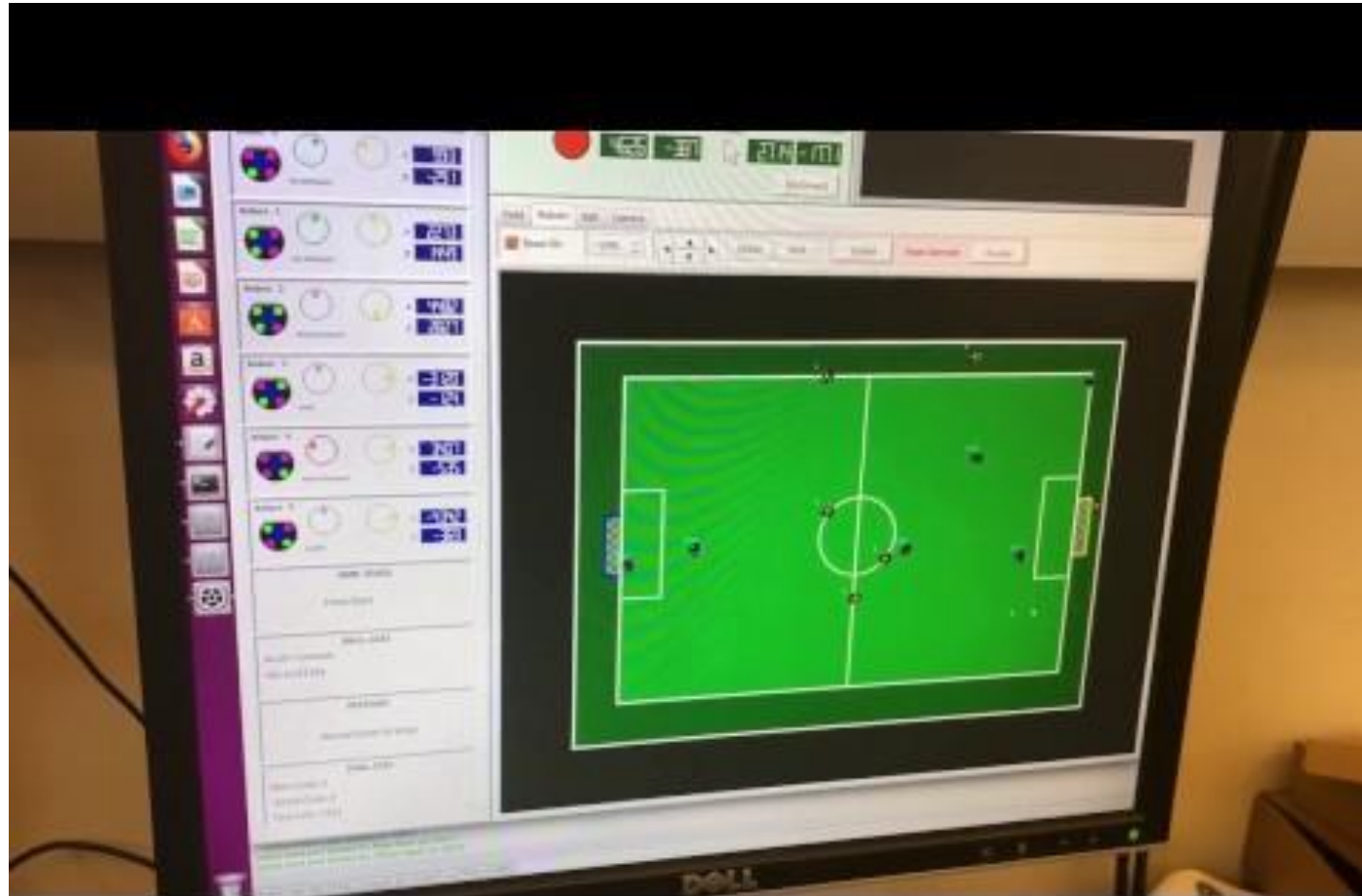
DEMONSTRATION OF SIMULATION



DEMONSTRATION OF SIMULATION



DEMONSTRATION OF SIMULATION



CHALLENGES

- Unsure of where to begin
- Lack of proper documentation
- Outdated and legacy code
- Incorrect Documentation (for simulation, documentation referred to a function that does not exist..)
- Codebase too big to understand

ISSUES THAT NEED TO BE FIXED

❑ SSL Refbox

- Current version of the refbox provided by the robocup is incompatible with our AI.
 - Glitches and crashes due to incorrect commands sent by the AI
 - Sends incorrect commands to robots
 - Sends multiple commands to robots which confuses the robots and the robots start spinning uncontrollably
- Older versions of the refbox no longer work with our AI due to
 - deprecated libraries
 - discontinued support from RoboCup
- Communication Layer
 - Refbox communication protocol (using protobuf) was changed recently. We need to update our AI to make it work on port 10003 instead. ([see](#))

SUGGESTIONS

❏ Ball Control

- Currently, The ball itself cannot be moved if it goes out of the field in RoboBulls.
 - Workaround is to override a robot and use it move the ball.
- In RoboCup, humans move the ball.
- It will be useful to create a function to just drag the ball (with a mouse) back into the field when it goes out

RESOURCES

- Robocup website (https://wiki.robocup.org/Small_Size_League/Rules)
- SSL Refbox (<https://github.com/RoboCup-SSL/ssl-refbox>)
- GrSim (<https://github.com/RoboCup-SSL/grSim>)
- The original AI code on Github (<https://github.com/mllofriu/robobulls2>)