

User Manual

April 29th, 2022



Team Code Duckies

Team Members: Anthony Simard, Ari Jaramillo (lead),

Daniel Rydberg, Chris Cisneros, Jacob Heslop

Sponsor: Dr. Truong X. Nghiem

Mentor: Vova Saruta

Table of Contents

| | |
|---|-----------|
| 1. Introduction | 3 |
| 2. Installation | 3 |
| 3. Configuration and Daily Operation | 4 |
| Move Blocks | 6 |
| Untimed Move Blocks | 6 |
| Other Blocks | 6 |
| Stop | 7 |
| Distance | 8 |
| Encoder Blocks | 8 |
| Creating and Running a Program | 9 |
| 4. Maintenance | 11 |
| 5. Troubleshooting | 11 |
| 6. Conclusion | 12 |
| Appendix: Useful Links | 13 |

1. Introduction

This manual is designed to assist in the installation and usage of the Code Duckies visual programming tool for Duckietown robots. This tool was developed and tested using a Duckiebot DB21M, and no fitness for purpose can be guaranteed for other models. This document will assume a DB21M is the model being used. This program in its current form allows for easy setup and usage with the potential for future improvements to utilize more of the bot's hardware.

2. Installation

Usage of this tool presupposes that you have a properly configured Duckiebot DB21M. The instructions for how to accomplish this may be found [here](#). You must complete this manual through at least "Unit C-6 - Handling - Duckiebot DB21, DB21M" along with "Unit C-19 - Operation - Networking." When connecting to a wifi network, we suggest choosing something easy to connect to such as a phone hotspot or personal router. Things like university wifi and apartment wifi usually require more configuration. We recommend that the individual setting up the bots follows the entire manual excluding Part D at least for the first bot to ensure they fully understand configuring and handling their bot. Part D is not relevant to the usage of this tool but is relevant to the normal usage of the bot without this tool. "Unit C-13 - Calibration - Wheels" is also recommended if you want the bot to move as accurately as possible. As is measuring and setting the wheel radius as described [here](#).

NOTE: This setup of the bot requires Ubuntu (or an Ubuntu-based OS), but usage of the tool does not.

NOTE: Before doing a unit, make sure you have done the prerequisites noted as "Requires: X" at the top of the page"

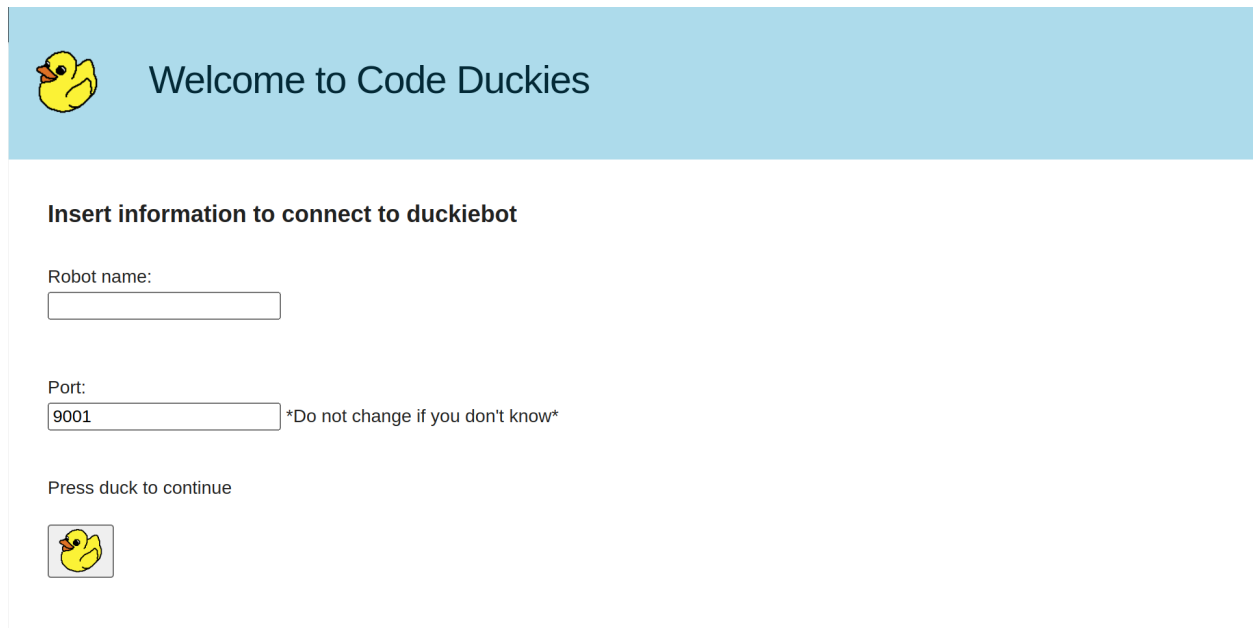
NOTE: The bot will be configured with both a name and a wifi network to connect to. These are both relevant pieces of information that we suggest noting down somewhere. The bot name, and perhaps the network as well, should probably be written on a sticker or piece of tape that is attached to the bot for easy reference.

Installation of the Code Duckies visual programming tool is exceedingly simple. One must simply clone the Code Duckies GitHub repository (found [here](#)). No dependencies should be necessary other than the properly configured bot.

3. Configuration and Daily Operation

When you want to actually use the tool, first ensure that the computer you intend to control the bot from is connected to the same wifi network as the bot. The bot should have been configured to connect to a particular network in Unit C-19 above, and the computer should be connected to this same network. Then open the index.html file found in the Code Duckies repository in a web browser. We suggest using Google Chrome or another Chromium-based web browser such as Edge, Brave, or Chromium itself. The tool did not work properly in Firefox in our testing, and it is doubtful that it would work in Internet Explorer. It has been most robustly tested in Chrome and Brave.

If you're at this point, you should be looking at the connection page of the tool (Fig. 1) which is prompting you to enter the name of your bot and click the duck button to continue. The name of the bot will have been given during the installation above. If you went through the manual, you may have noticed that sometimes they want you to enter "name.local" not just the name. As they explain, this is a quirk of many routers. We suggest you enter "name.local" first, and then try the name without the .local if that does not work. You can also enter the IP address of the bot if you know it, but in most cases, the name will be easier.



Welcome to Code Duckies

Insert information to connect to duckiebot

Robot name:

Port: 9001 *Do not change if you don't know*

Press duck to continue

Figure 1: Connection page

NOTE: The port 9001 came from "ARG PORT" in [this](#) file, if things really aren't working check to see if this has changed. Otherwise, the port should not be changed. This is set by the core Duckietown development team, not us, but it has not changed since Duckietown was released.

After connecting to the bot, you should see the actual programming page (Fig. 2). This page contains the toolbox on the left with the start/stop button above it. The programming area in

which to assemble a program out of blocks from the toolbox in the center, and a menu on the right that contains save, load, new program, and reset connection functionality.

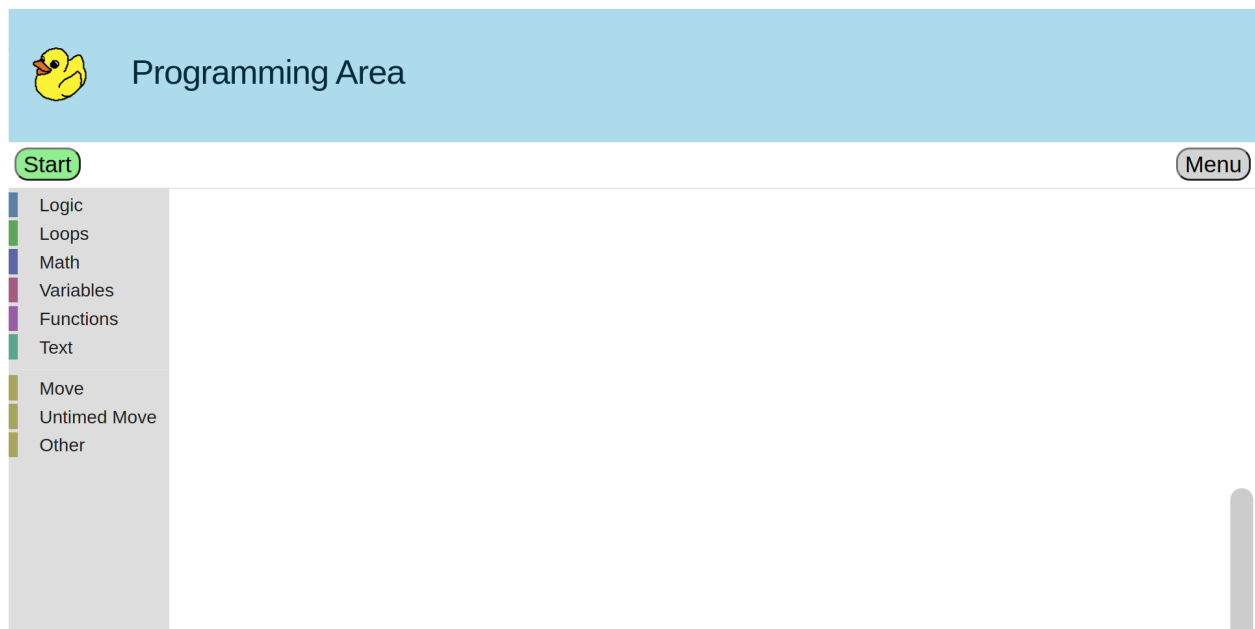


Figure 2: Programming page

Everything above the white dividing line in the toolbox is provided by Blockly. These are various primitives (loops, if/else, variables, etc.) that are necessary to the construction of programs. If there is any confusion about these blocks, please consult the Blockly documentation. This manual will document the functionality of the custom blocks currently included in the tool that may be found beneath the white dividing line. Note that there is significant potential for new blocks (particularly blocks using the camera for lane following and obstacle detection/avoidance). The blocks described here are what is included in the tool as of the time of writing this document.

NOTE: All times are in seconds, the distance is in meters, the linear velocities are in meters per second, and the angular velocities are in radians per second. Also, note that these are approximate. A properly calibrated bot with a low latency connection to the computer controlling it will be more accurate than a poorly calibrated bot on a higher latency connection.

NOTE: Linear velocity ranges from -0.4 to 0.4 and radial velocity ranges from -8 to 8 . These limits are based on the min and max values sent to the bot via the built-in virtual joystick keyboard control.

NOTE: The bot pivots around its central axis. This means a radial velocity of 1 on the turn left block will cause the right wheel to turn forward at 1 radian per second and the left wheel to turn backward at 1 radian per second.

Move Blocks

The blocks in the “Move” section of the toolbox (Fig. 3) allow the user to move the bot at set speeds for set amounts of time. They allow for the most basic pre-scripted movement of the bot. For example, you could use these blocks in conjunction with the Blockly built-ins to make the bot move in pre-defined shapes or follow pre-defined routes.

The “move bot” block allows the user to make the bot move forward/backward and turn left or right at the same time. The other blocks allow for movement in the specified direction only.

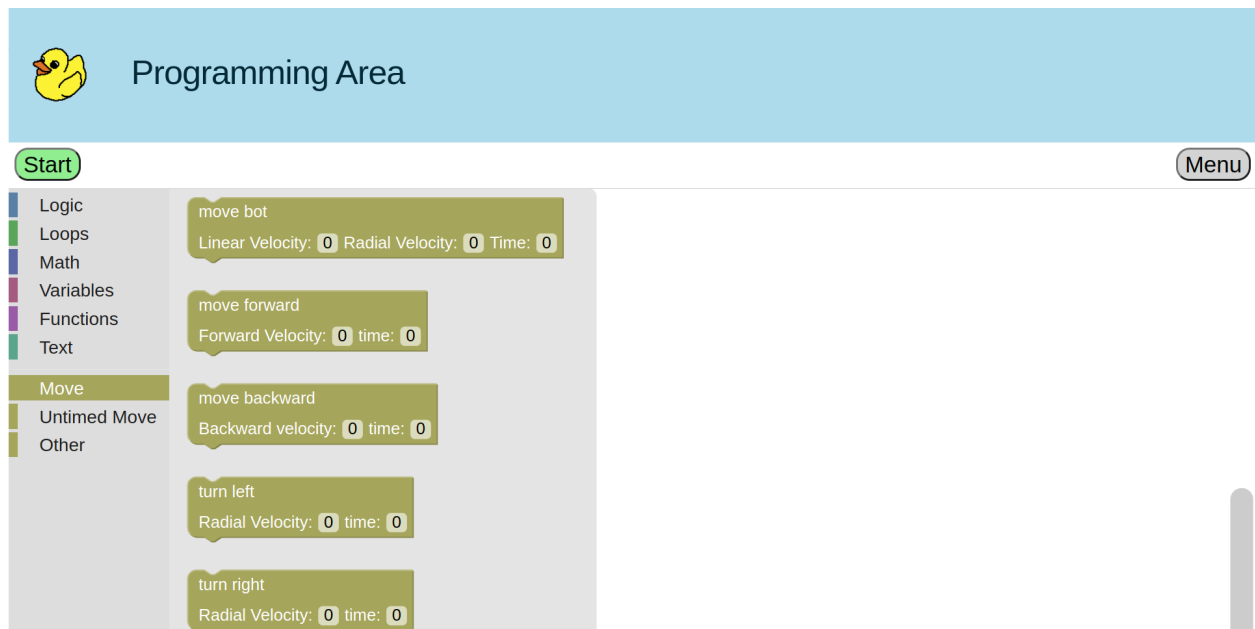


Figure 3: Movement blocks

Untimed Move Blocks

These blocks (Fig. 4) are functionally equivalent to timed move blocks with a time of 0. They are intended to be used in more complicated programs than the timed move blocks. For example, if you wanted to make a program that made the bot move forward at a set speed until something was within 10 centimeters of the front of the bot then stop, you don't care about the time, you just want it to move forward.

Other Blocks

These blocks (Fig. 5) are a bit more complicated than the movement blocks and require a bit more explaining in detail. With the exception of the “stop” block, they are used to operate the sensors on the bot. In our current model, this is where things like camera and IMU sensor reading and usage would go when/if implemented, but at that point, it might be useful to further split up the toolbox.

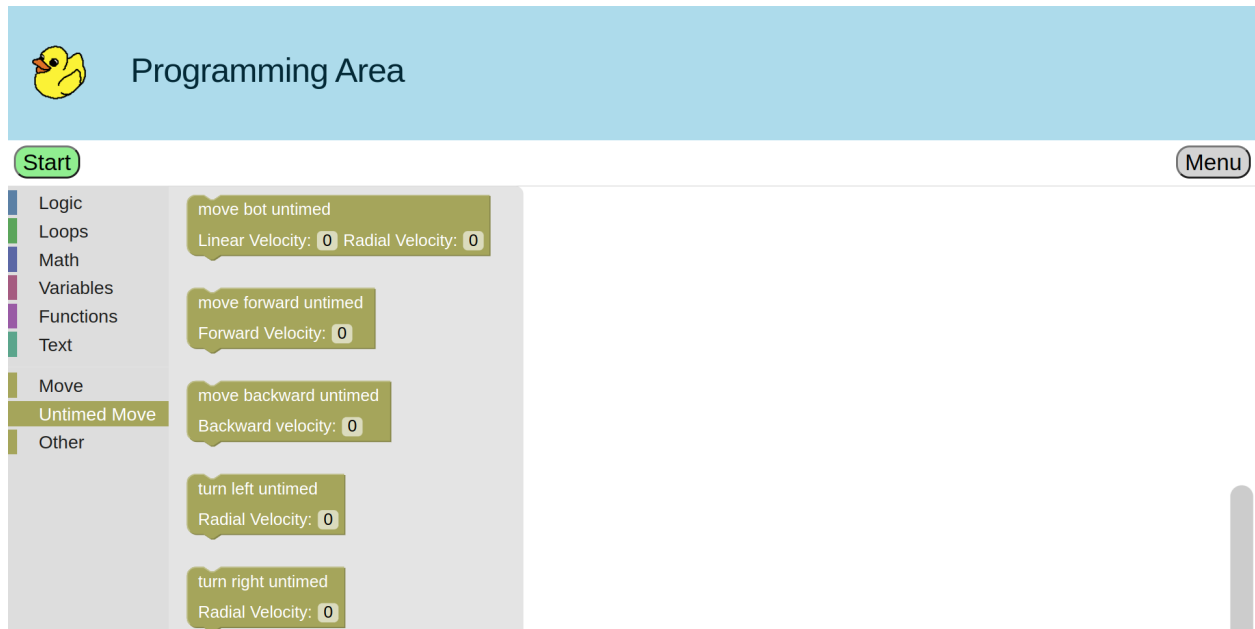


Figure 4: Untimed movement blocks

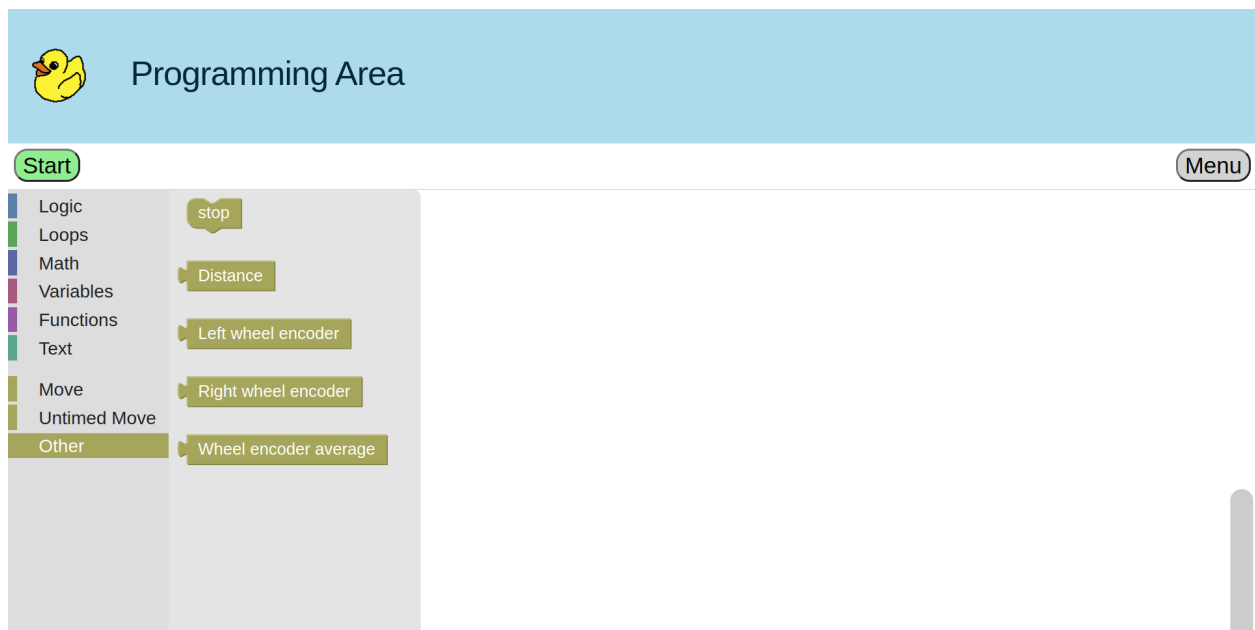


Figure 5: Other blocks

Stop

This block is semantically equivalent to any move block with zero velocity. It just sends a zero linear and zero radial velocity command to the bot.

Distance

This block gets the distance to the nearest obstacle in front of the bot from the front time of flight sensor. It allows for very basic object detection and avoidance by simply stopping or turning when there is something in the way. More advanced versions of this will be possible when/if the front-facing camera on the bot is integrated.

Encoder Blocks

These get the current reading from the wheel encoders on the bot for the left and right wheel respectively. The average is the sum of the two measures divided by two and then rounded to the nearest whole number. The behavior of the wheel encoders is described [here](#).

These values should usually be stored in Blockly variables and manually updated/accessed as demonstrated in Figure 6. Note that this is different from the distance sensor data which generally does not need to be stored in a Blockly variable. This is because with the distance data we are usually interested in the current measurement whereas with the encoder readings we are usually interested in the change over time

Note: The variable names `__DISTANCE__`, `__LEFT_TICKS__`, and `__RIGHT_TICKS__` are reserved and should not be used for Blockly variables.

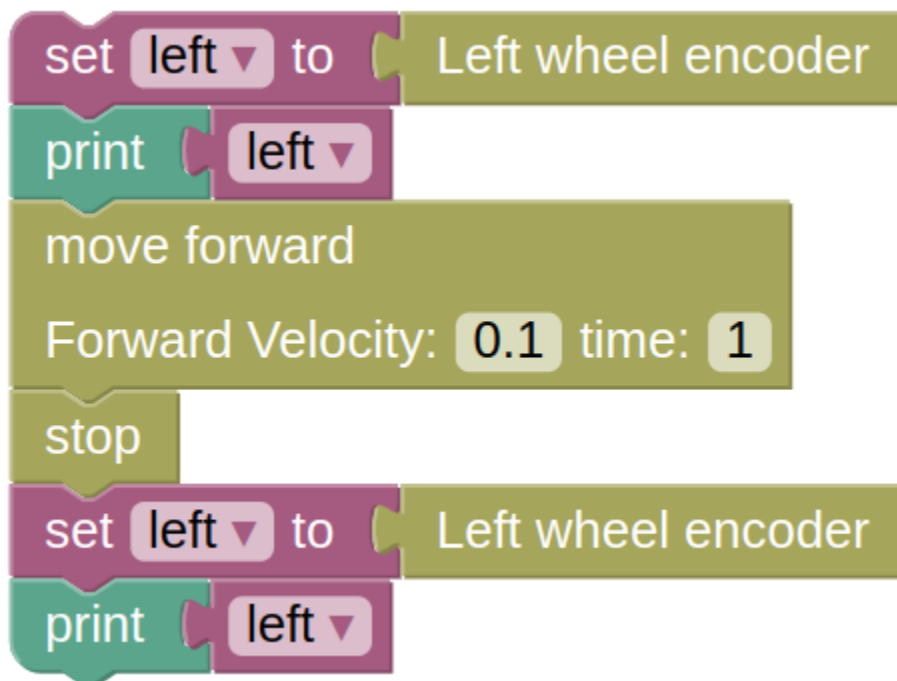


Figure 6: Set the Blockly variable “left” to the measure from the left wheel encoder and display the value then move the bot forward at .1 m/s for 1 second and stop it then read the value and print it again.

Creating and Running a Program

Once a program has been assembled out of the blocks, the user should click the start button to run the program on the bot. If everything is connected correctly, the bot should begin executing the program. Upon clicking the start button, it will be replaced by a stop button. Clicking this stop button at any point during the program execution will halt the bot and turn the button back to a start button. If the program ends on its own, the bot will stop moving and the start button will change back to a start button.

Note: Upon exit of a Blockly program either naturally or via the stop button being clicked, the bot is sent two messages. One is identical to the message sent by the “stop” block. The other is a failsafe that sets a flag on the bot that prevents it from moving at all until the flag is unset regardless of what commands it is sent. The start button unsets this flag to allow the bot to move again during program execution. This means that if you are using this tool, then try to move the bot by other means external to this tool, those other means will not work unless you unset the flag. The flag is set by sending a “duckietown_msgs/BoolStamped” type message with data “true” on the “/`<name>`/wheels_driver_node/emergency_stop” topic, and it is unset by sending a “false” on the same topic. Note that this is the flag that is toggled by pressing the “e” key when using the built-in keyboard control described [here](#). The flag is also unset by default when turning on the bot.

A key thing to understand about the execution of these programs is that the Duckiebot will continue to execute the last move command sent to it until it receives a new one. If the user clicks stop, or the program ends naturally, the bot will receive a command to stop moving. This means that if you have a program like Figure 7, the bot WILL NOT stop when the distance sensor reads less than 30cm because it will continue executing the move forward at a speed of .1 it was last sent. If you have the program in Figure 8, the bot will stop moving due to hitting the stop block. If you have the program in Figure 9, the bot will stop because the program will exit.

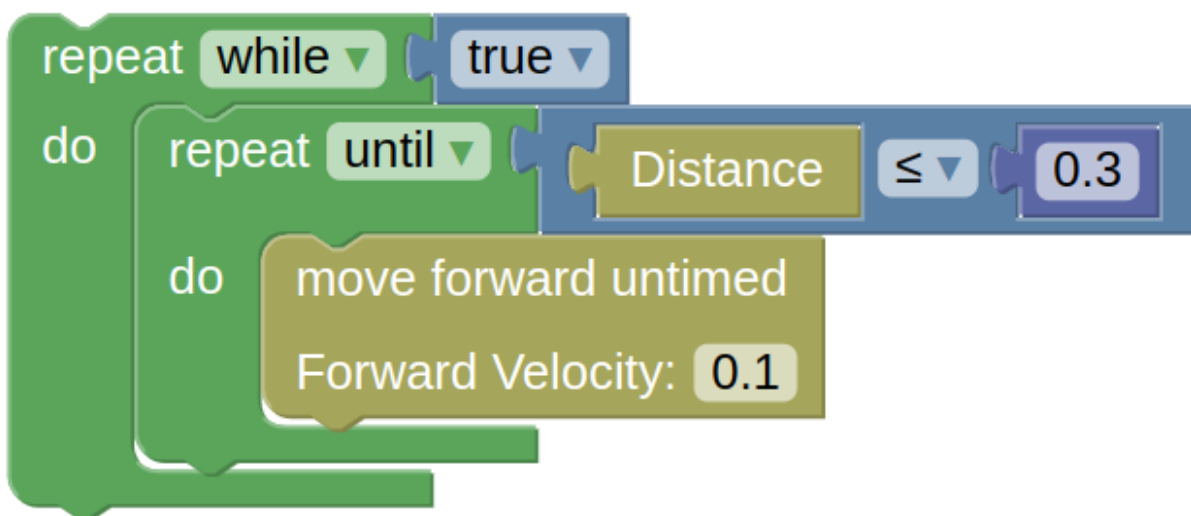


Figure 7: The bot will continue executing the last move forward it received and not stop

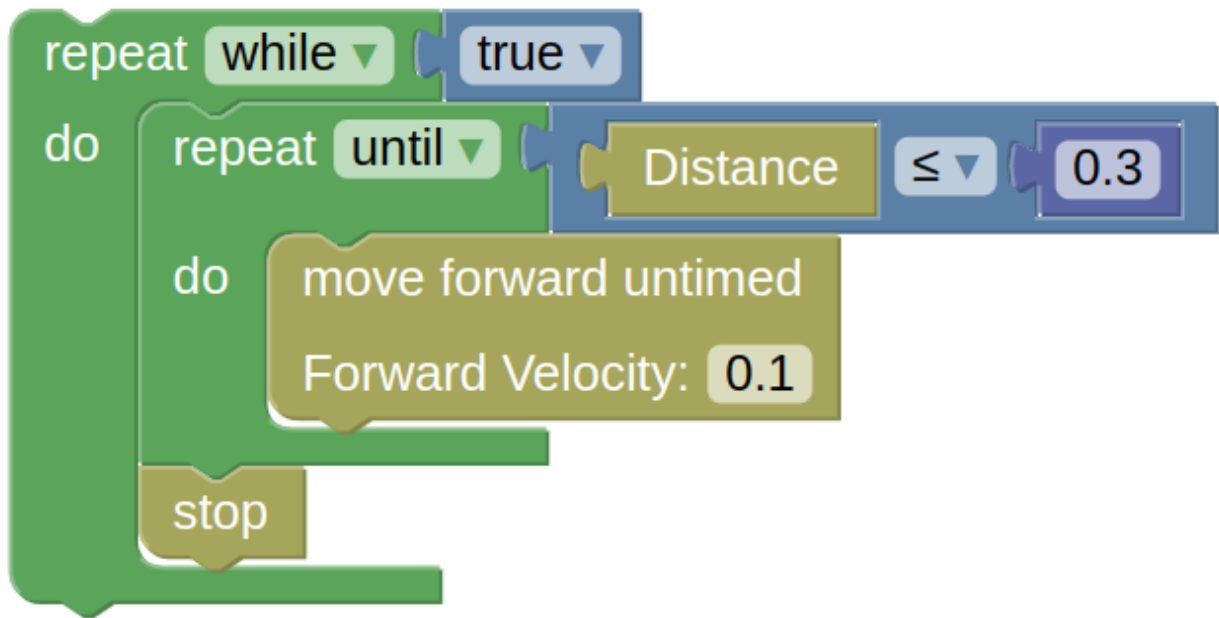


Figure 8: The bot will be stopped by the “stop” block

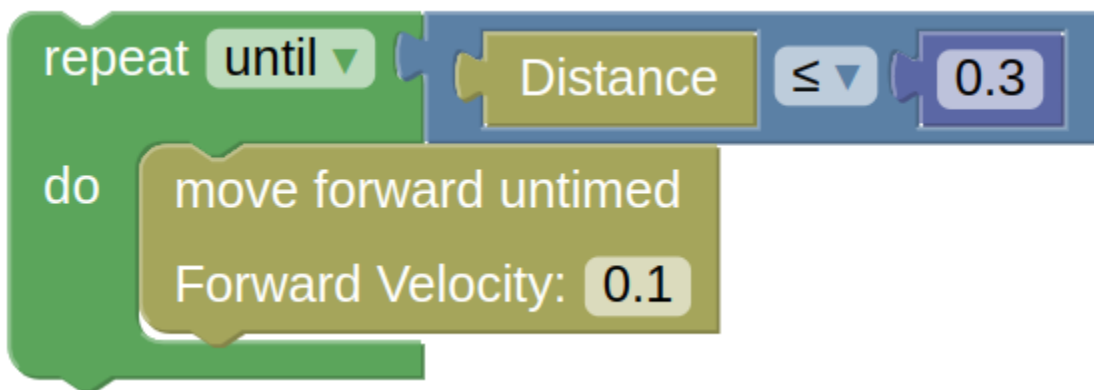


Figure 9: The bot will stop when the program exits

NOTE: There is some information, including the actual JavaScript the tool is executing on the back end of the blocks, logged in the console. If your program is not behaving quite how you expect, and you are comfortable with JavaScript, feel free to read that output.

Moving onto the menu on the right side of the screen (Fig. 10). The buttons in the menu work as follows:

1. **Save Project:** This will download a .txt file containing xml describing your visual program. Depending on how you have configured your browser, it may prompt you to name the file, or it may simply put it in your downloads folder with a default name.

2. **Load Project:** This will open your file explorer and prompt you to load a project that you previously saved. If you attempt to load an invalid file, it will simply not work and alert you to the fact that the file was invalid.
3. **New Project:** This will prompt the user to type either “yes”, “y”, or “yeah” into a box asking if they are sure they want to clear their workspace. If they do so, the programming area will be cleared of all blocks. If you have a project you like, save it before doing this.
4. **Reset Connection:** This will take the user back to the connection page allowing them to connect to a new bot, or attempt to connect to the same bot again if they failed last time.

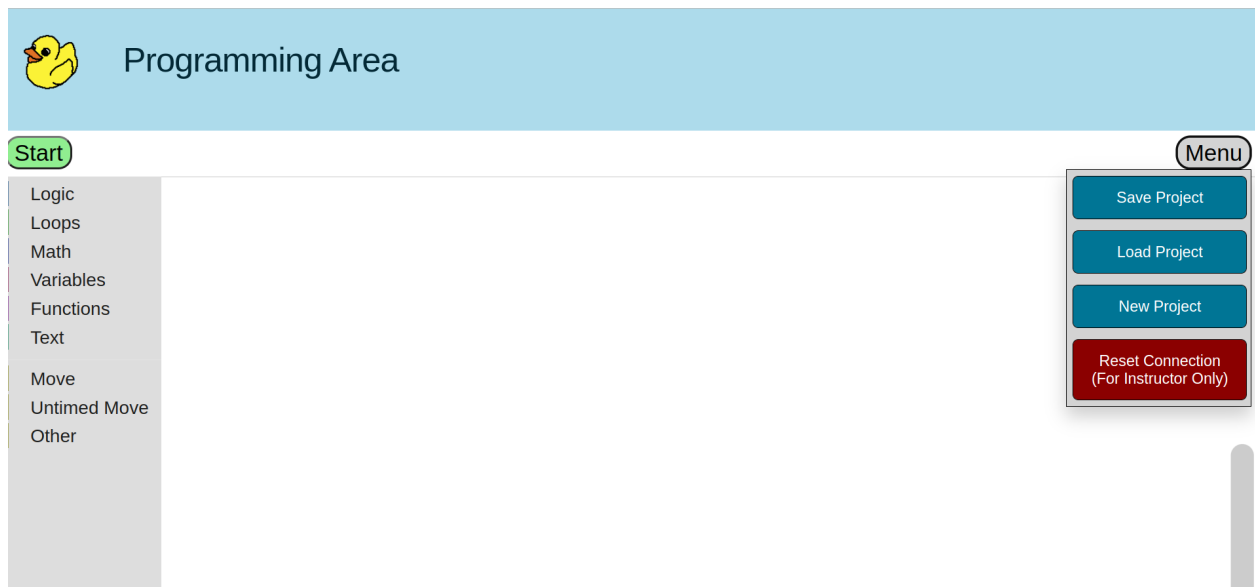


Figure 10: Menu

4. Maintenance

There is not really any maintenance to be done to keep this product working unless significant changes are made to Duckietown that break this tool. If modifications to the code are necessary to either accommodate future Duckietown changes or to add functionality to the code our GitHub repository is public and, as mentioned before in the installation section, may be found [here](#).

5. Troubleshooting

The most frequent issue encountered in testing was a failure to connect to the bot. This can be for a number of reasons. We will list some troubleshooting steps we found helpful in our testing below.

1. Make sure you entered the bot name correctly. A typo in the bot name will prevent things from working.
2. If you double-check the bot name and it still does not work, make sure you can ping the bot (open a terminal and type “ping <name.local>”). If that works you will get a response that looks something like Figure 11. If you wait a while and get nothing, you are unable to connect to the bot.
3. If you are unable to connect to the bot go through the following steps attempting to ping the bot in between each until the problem is resolved. If none of these resolve the issue, you have likely encountered something we did not.
 - a. Make sure your computer is connected to the correct network
 - b. Reboot the bot
 - c. Disable the firewall on your computer (how to do this is heavily dependent on what OS you are using, please look into how to disable the firewall on your OS)
 - i. If this works, you might want to look into creating a rule to allow two-way communication with the bot (again how to do this is very dependent on your OS and your firewall), so you do not need to leave your firewall off entirely
 - d. Go back to [Unit C-19](#) in the documentation and make sure you have correctly configured the connection to the wifi you are attempting to connect to
 - i. Again, we suggest using something that is simple to connect to such as a phone hotspot. They only require that the bot know the name of the wifi and the password (or you can leave them open) instead of requiring extra steps like many university or apartment routers.

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.083 ms  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.086 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.103 ms
```

Figure 11: Successful ping

If there are any issues with the tool itself, start by refreshing your web page. If that doesn't work, close the tool and re-open it in a new tab.

6. Conclusion

We hope this product is at least a satisfactory start to the project and will be used and expanded upon in years to come. Best wishes from the Code Duckies development team Ari Jaramillo, Anthony Simard, Chris Cisneros, Daniel Rydberg, and Jacob Heslop.

Appendix: Useful Links

These are some potentially useful links that are up to date as of the time of writing this document in April 2022.

Code Duckies Repo: <https://github.com/cloudandr0id/Code-Duckies>

Duckietown GitHub Organization: <https://github.com/duckietown>

Duckietown Documentation: <https://docs.duckietown.org/daffy/>

Blockly Documentation: <https://developers.google.com/blockly>