

Welcome to VEX Superquest 2019 The Dalles



Introductions

- Take 5 minutes to meet your neighbors (All sides)
 - Name
 - School or Affiliation
 - Robotics Programs (VEX, FRC, FTC, FLL, SkillsUSA, ...)
 - One thing they would like to get from this workshop.
 - Something else about them. Hobbies, distance travelled, summer plans, ...
- Participants introductions

Schedule: Best Guess

Tuesday

- Course Overview
- 9:00 am Session 1
 - Introductions
 - V5 System Overview
 - Programming the V5 Brain (no motors)
 - **Face/Robot Activity**
- 10:45 am Session 2 10:45
 - Coding: The while loop and Sensors
 - Using Sensors with V5
 - Bumper Switch/Limit Switch/Touch Sensor
 - Ultrasonic Range Finder
- 12:00 Lunch and Learn: Andrew Scholer: OCSTA
- 1:00 Tuesday Session 3:
 - Using Sensors with V5 Continued
 - Potentiometer
 - Output to the Remote
 - Line Tracking Sensors
- 2:45 Build Time: V5 Clawbot
- If time: Using Functions in Programming
- 3:45 pm Exit Survey
- 4:00 pm End of Day 1.
 - Lab will be open for work after 4:00

Wednesday

- Welcome Back
 - Go over exit survey/Questions
- Wednesday Session 1: Variables/Loops
 - Sentry Simulation
- Wednesday Session 2: Intro to Sensors:
 - Bumper Switch
 - Sentry Simulation with roadblocks
 - RoboMower with Touch (Bugbot)
 - Atlas Stone Competition: Blocks on boxes
- Lunch and Learn Terrel Smith
- Wednesday Session 3: Remote Control
 - Drive Train
 - Slalom/Soccer/
 - Turn Button Challenge
 - Arm/Claw
 - Pick up challenge/ Urban Search and Rescue
 - Competition Template
- Wednesday Session 4: Additional Sensors:
 - UltraSONIC Range Finder
 - Line Tracking Sensors
- Exit Survey

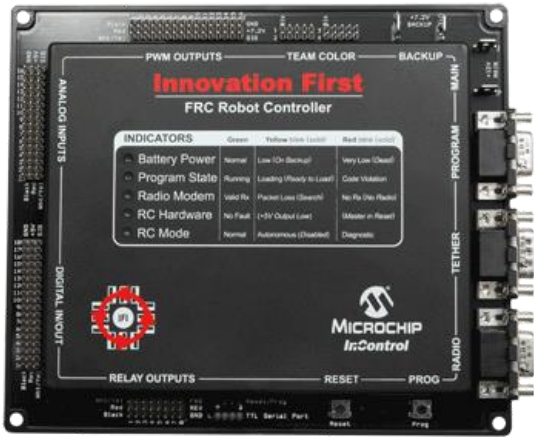
Thursday

- Welcome Back
 - Go over exit survey/Questions
- Thursday Session 1: Additional Sensors (Cont. from Wed.)
- Thursday Session 2: Competition Template
- Lunch and Learn: Greg Smith (Project Management)
- Thursday Session 3: In class Practice Tournament
 - Autonomous
 - Driver Operator
 - Alliance Selection (If time)
- Thursday: 3:00
 - End of conference survey
 - Clean up
 - Thanks for coming and I hope to see you this season.

VEX V5 Overview



A little History



2000
IFI FRC Robot Controller



2003
IFI EDU-bot



2006
PIC Microcontroller



2010
Cortex Microcontroller



V5 Robot Brain

- 4.25" (480x272 pixels) (480x240 programmable) full color touch-screen
- Dashboard provides real-time diagnostics
- 21 smart ports: Motors, Smart Sensors, Radio, or Tether in any port.
- Eight 3-wire ports for using past sensors: Digital or Analog sensors.
- Built in programs
- microSD Card Expansion. FAT32 up to 16 GB
- Automatic Wire Check
- Automatic device firmware check
- 128 Mbytes Ram
- 32 Mbytes Flash
- Wireless: VEXnet 3 and Bluetooth 4.2



V5 Wireless Controller

- LCD Screen for real-time information
- Start and stop programs from the controller
- Programmable haptic (force) feedback.
- Programmable monitor
- Competition practice mode – synch up with other robots and run practice matches.
- Built in VEXnet 3.0 and Bluetooth
- Integrated rechargeable battery



V5 Robot Battery

- Lithium-Ion 1000mAh
- 12.8 Volts
- Built-in indicator lights
- recharges in about an hour
- 100% power at low voltage (51% for 7.2V)
- Lifespan +/- 2000 full charges (500 for 7.2 V)
- 256 Wats (58 W for 7.2 V)
- 20 Amps max (8 Amps on 7.2 V)
- Separate connections for charging the battery and connecting to the robot. Makes it easier to charge the battery while it is on the robot.



V5 Robot Radio

- VEXnet 3.0 supports 500 simultaneous robot channels.
- Capable of wirelessly connecting to Bluetooth (4.0 and higher) enabled devices
- LED indicator for linked, scanning and active modes.
 - Solid Red:
 - No radio communication
 - Flashing Red:
 - Active connection between V5 Brain and V5 Controller
 - The V5 Controller is not plugged into the field control system OR the brain is not running a user program
 - Flashing Green
 - An active connection between the V5 brain and the V5 Controller
 - The V5 Controller is plugged into a Field Control System
 - The V5 is running a user program
- You can download programs wirelessly using the remote.



V5 Smart Motors



- Built in encoders!
- Comes with Green 200 RPM gear cartridge. (18:1)
- Cartridges with Red 36:1 (100 RPM) and Blue 6:1 (600 RPM) available
- Can use standard and high strength shafts
- Twice as powerful as the Cortex 393 Motors
- Can see the color of the cartridges from outside the motor. No need to guess on the speed setting for the motor.



Programming the Parts: Some of the Options

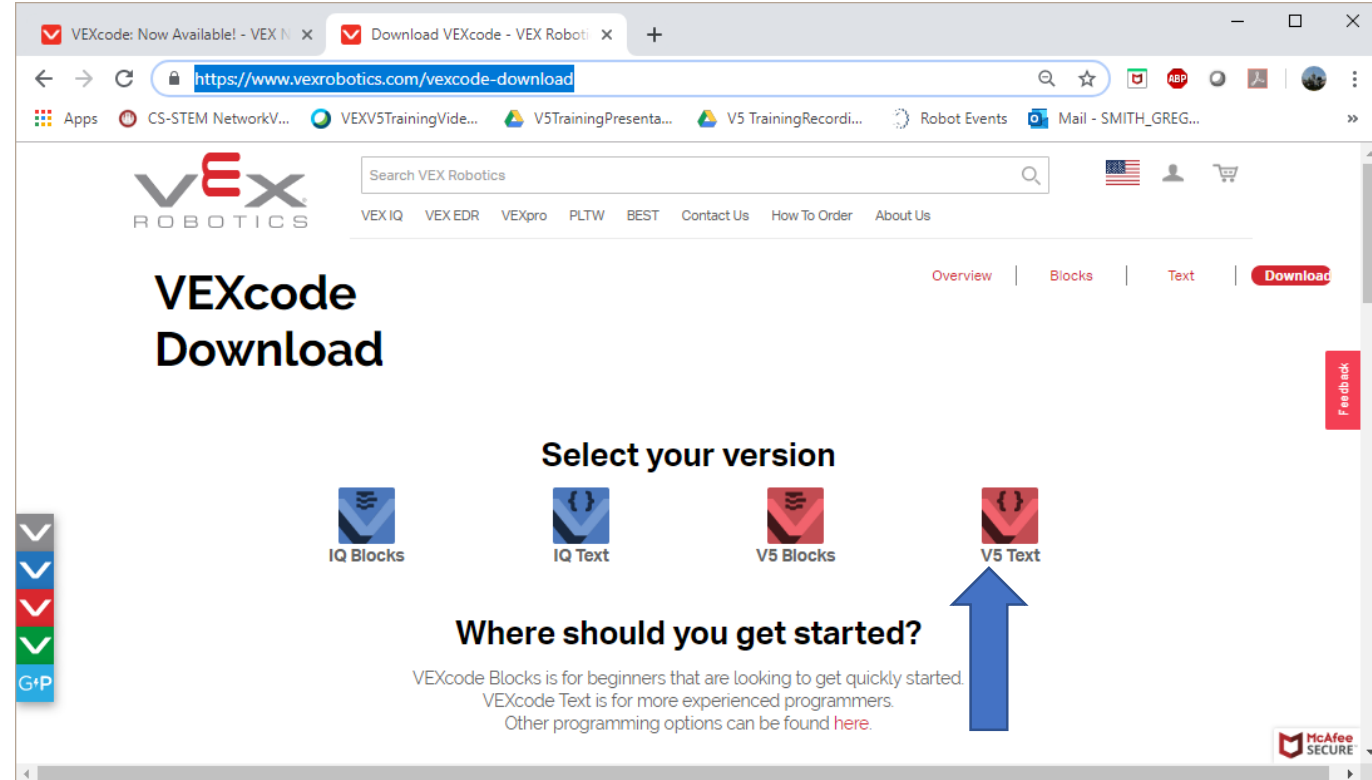
- VEX Coding Studio: Most commonly used last season
 - Free
 - MODKIT: Drag and Drop
 - VEX C++
 - C++ Pro
- Robot Mesh Studio: Not written by VEX
 - Free
 - VEX iQ, VEX EDR Cortex, VEX V5 (Python), VEX V5 C++
- **VEXCode: The direction VEX is supporting**
 - Free
 - VEXcode Text (8/2/2019)
 - VEXcode Blocks (Due August 2019)
 - PC/ MAC and Chromebook versions
 - Similar to RobotC and VEX Coding Studio.

Future Features (Version 1.0 due in the next month or two)

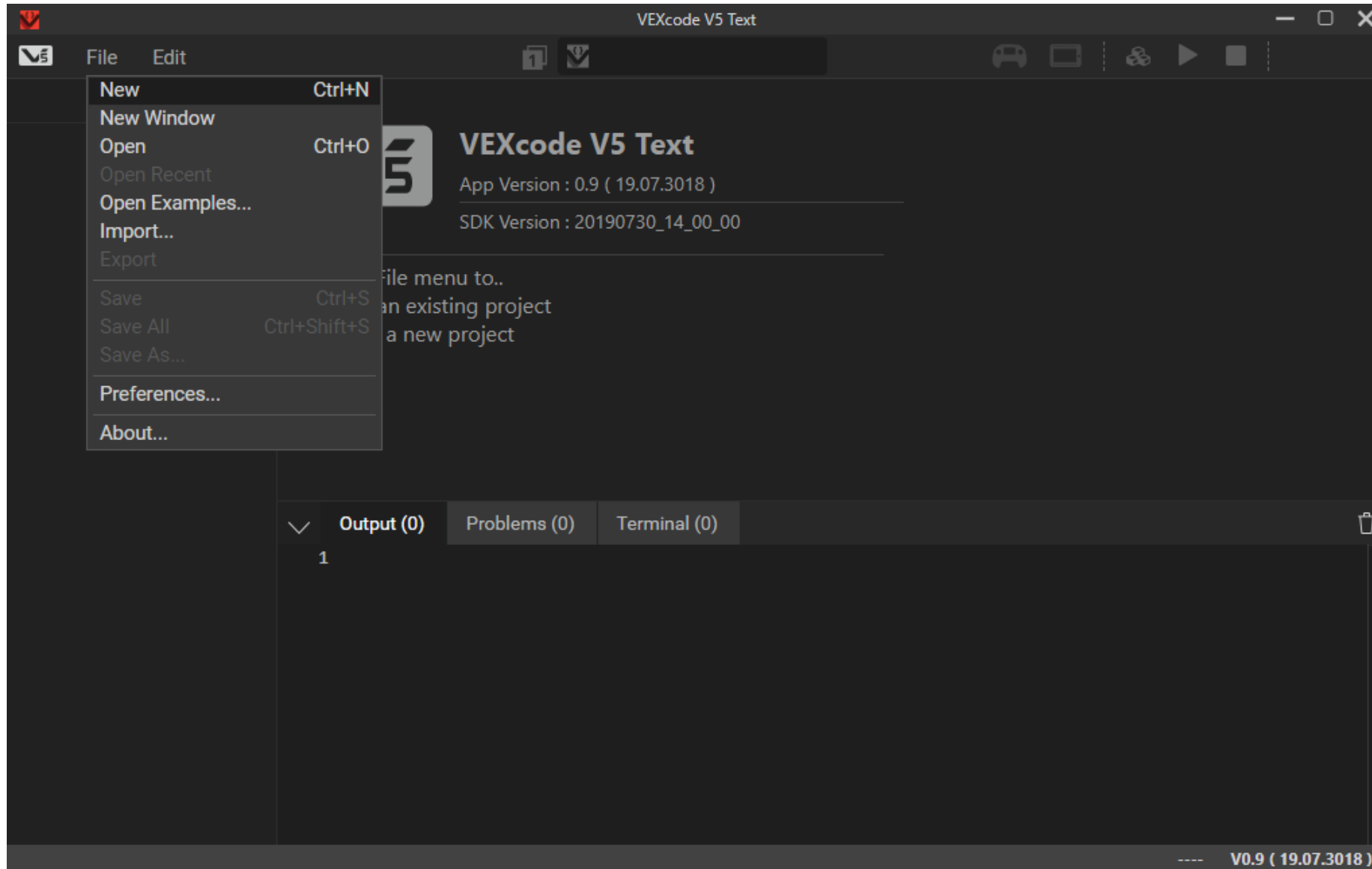
- Remove classes from commands
- Language help
- Graphical Device Manager
- Video Tutorials
- Spanish and Chinese languages

Getting Started with VEXcode V5 Text

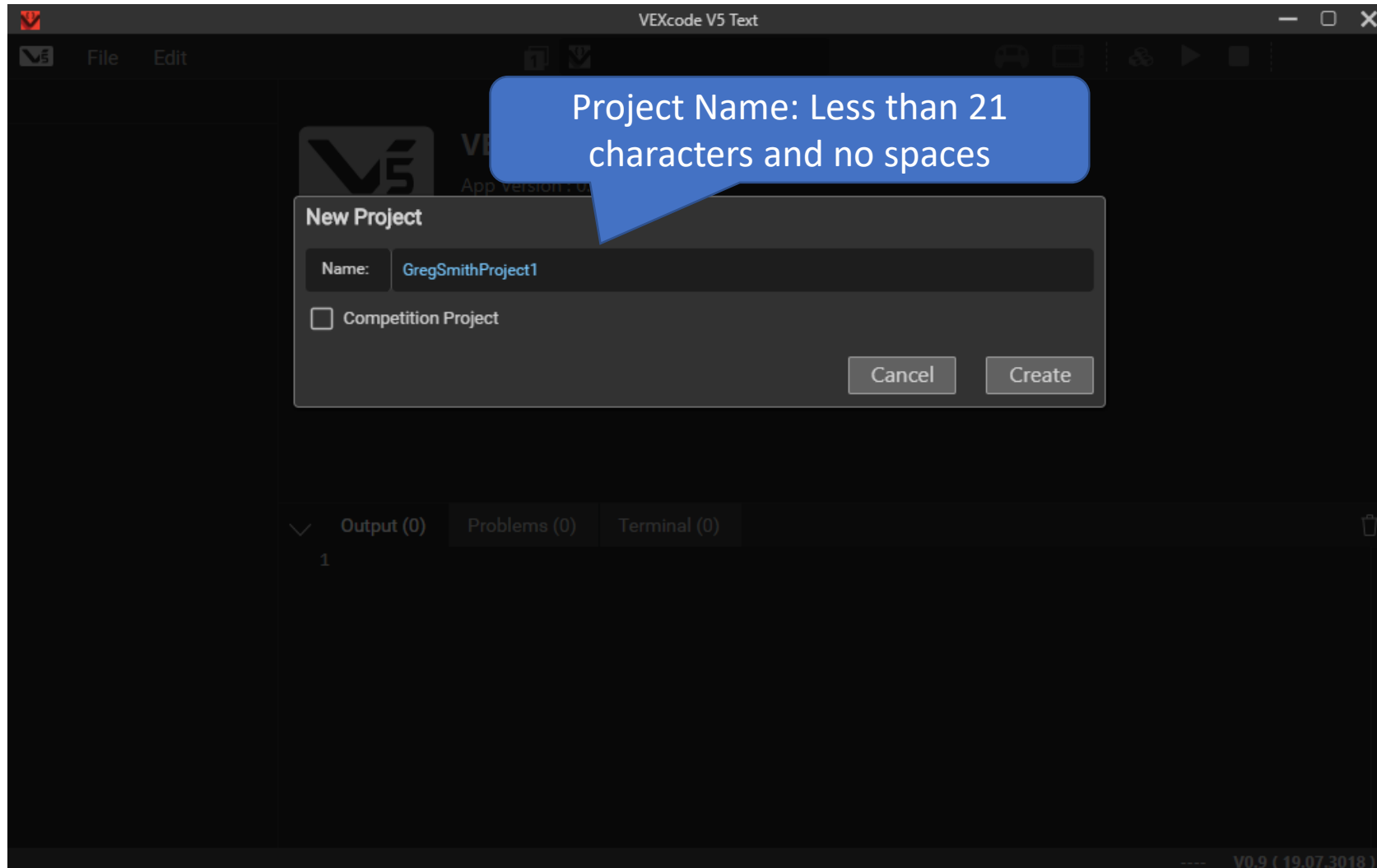
- Download and Install
 - <https://www.vexrobotics.com/vexcode-download>
- Open VEXcode V5 Text



Open the New Project Dialog Menu



Name it and Select the Type of Project



Editor

The image shows the VEXcode IDE interface with several callout boxes pointing to specific features:

- Device Info**: A box containing the text:
 - Brain Name
 - Team Name
 - Brain ID
 - VexOS
 - Firmware
- Compile and Download**: A box pointing to the download icon in the toolbar.
- Split Editor**: A box pointing to the split editor icon in the toolbar.
- Stop**: A box pointing to the stop icon in the toolbar.
- Play**: A box pointing to the play icon in the toolbar.
- Program Slot**: A box pointing to the program slot icon in the toolbar.
- Project Name**: A box pointing to the project name field in the toolbar.

The main editor area displays the following C++ code:

```
14 vex::brain Brain;|
15
16 // define your global instances of motors and other devices here
17
18
19 int main() {
20     int count = 0;
21
22     while(1) {
23         Brain.Screen.printAt( 10, 50, "Hello V5 %d", count++ );
24         // Allow other tasks to run
25         this_thread::sleep_for(10);
26     }
27 }
28
```

Editor Helping with Errors

5) Suggested correction

4) Click on Light Bulb for suggested correction

1) Highlights Error

2) Click on 'Problems' tab to get more info on errors.

3) Error info. Description and location (row 23 column 27) of error.

The screenshot shows the VEXcode V5 Text editor interface for a project named 'GregSmithProject1'. The main editor window displays a C++ file named 'main.cpp' with the following code:

```
16 // define your global instances of motors and other devices here
17
18
19 int main() {
20     int count = 0;
21
22     while(1) {
23         Brain.Screen.print At( 10, 10, "Hello V5 %d", count++ );
24         // Allow other tasks to run
25         this_thread::sleep_for(10);
26     }
27 }
28
29
```

The code contains several errors highlighted in red: a missing semicolon at the end of the while loop (line 27), a reference to a non-static member function 'print' (line 23), and an undeclared identifier 'At' (line 23). A lightbulb icon is visible next to the error on line 23. A context menu is open over the lightbulb, showing the option 'insert ';''. The 'Problems' panel at the bottom shows three errors:

- Expected ';' after expression (fix available) (23, 27)
- Reference to non-static member function must be called (23, 9)
- Use of undeclared identifier 'At' (23, 28)

Simple Program Break Down

```
12
13 // A global instance of vex::brain used for printing to the V5 brain screen
14 vex::brain Brain;
15
16 // define your global instances of motors and other devices here
17
18
19 int main() ← Program starts here
20 {
21     Brain.Screen.printAt( 10, 20, "The Dalles Loves February 2/15/2020!");
22 } ← Program ends here
23
```

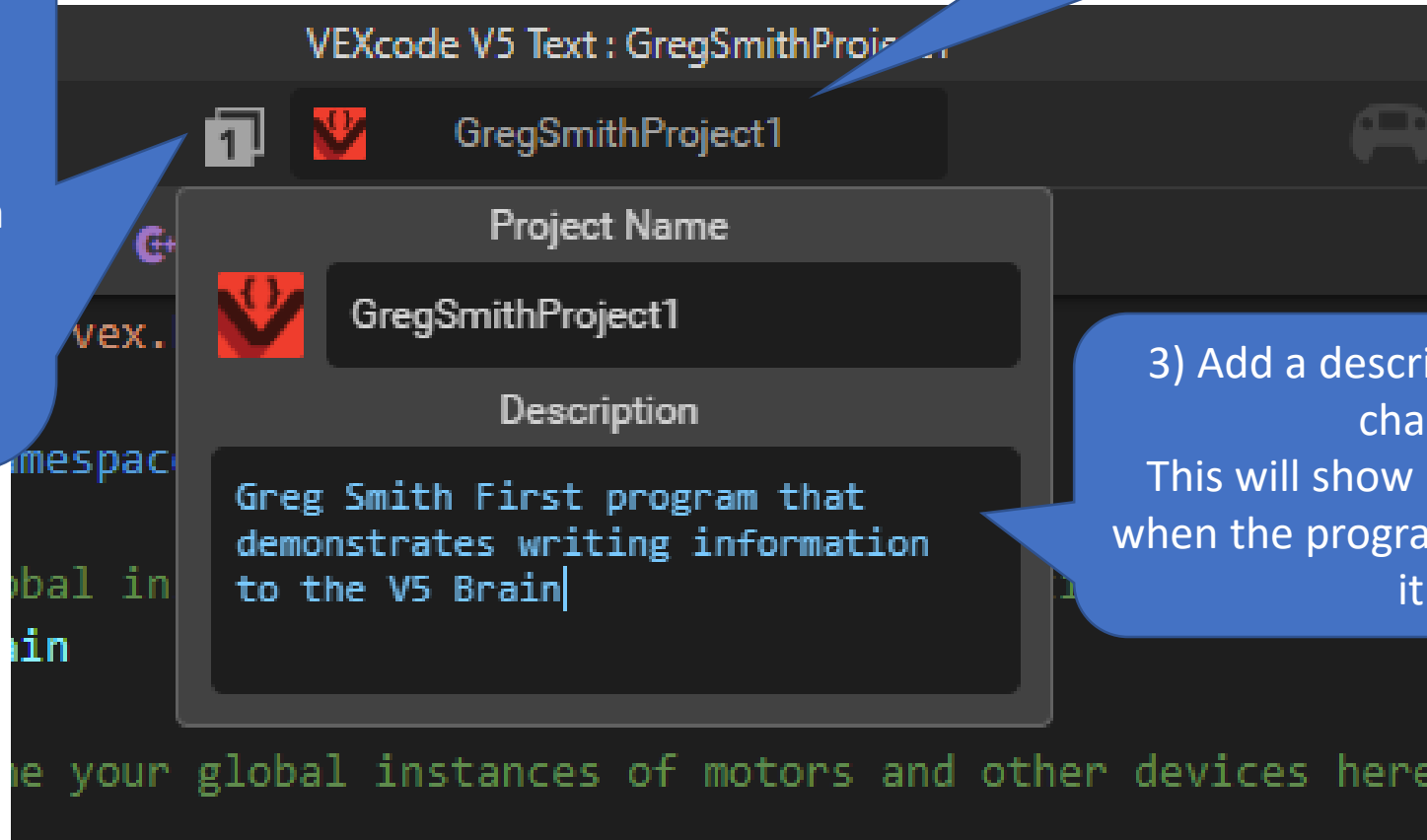
Robot “prints” message. Only one command in this program

Project Description (Notes)

1) The V5 Brain can store 8 programs in different slots. Currently Slot 1 is selected. Click on this to select a different slot as needed.

2) Click on the Project Name (GregSmithProject1 in this case.) to access the Project Description.

3) Add a description (80 or fewer characters)
This will show up on the V5 brain when the program is selected before it runs.



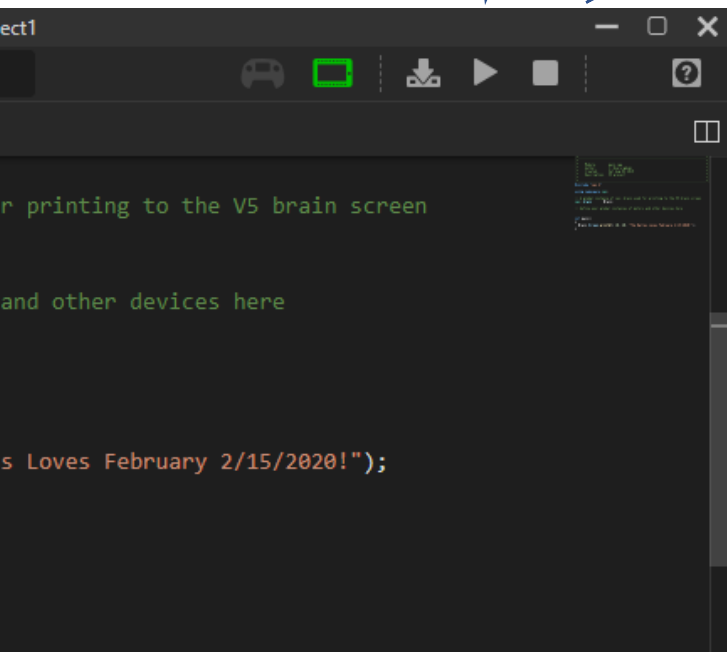
The screenshot shows the VEXcode V5 Text editor interface. At the top, it says "VEXcode V5 Text : GregSmithProject1". Below that, there is a tab labeled "1" and a red icon with a white "1" inside a square, representing the selected slot. The project name "GregSmithProject1" is displayed next to a red icon with a white "1" inside a square. Below the project name, there is a "Project Name" field containing "GregSmithProject1" and a "Description" field containing "Greg Smith First program that demonstrates writing information to the V5 Brain". The background shows some code snippets like "vex.", "amespace", "lobal in", "in", and "e your global instances of motors and other devices here".

Download and Run the Program

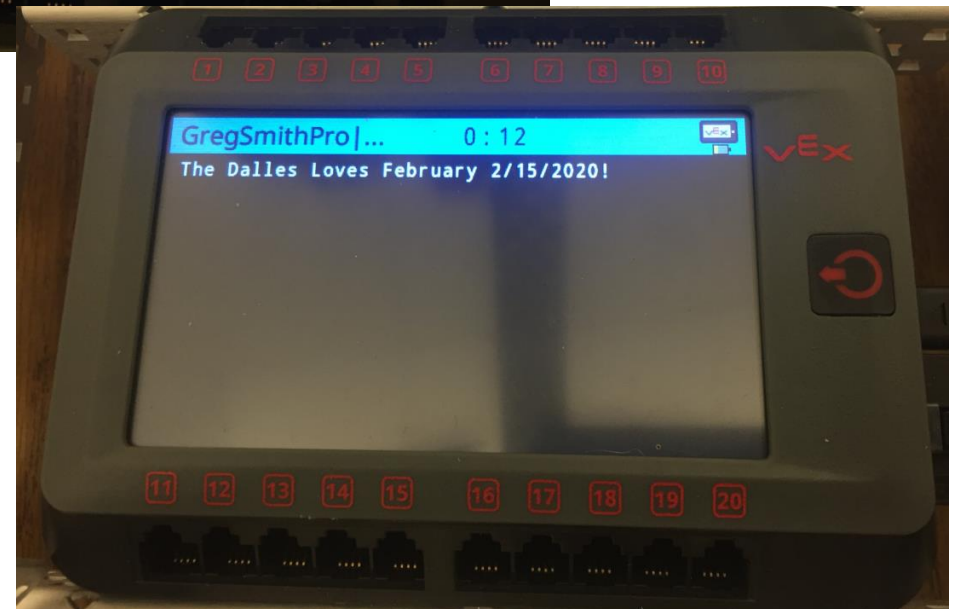
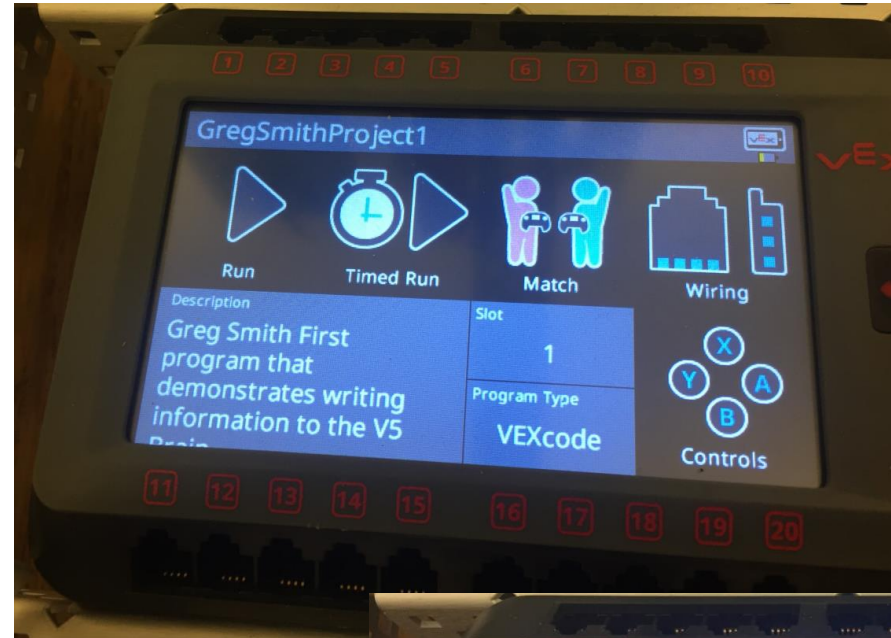
1) Download

2) Run

3) Stop



```
... printing to the V5 brain screen  
... and other devices here  
... Loves February 2/15/2020!");
```

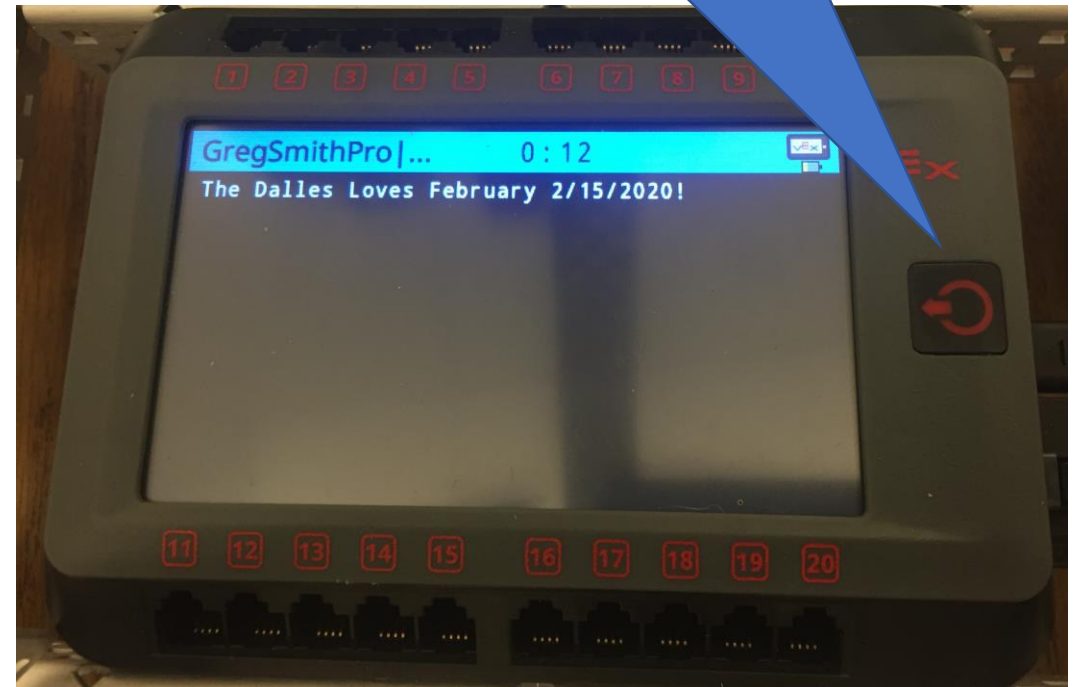


If you just downloaded the program: You can Run the Program from the V5 Brain

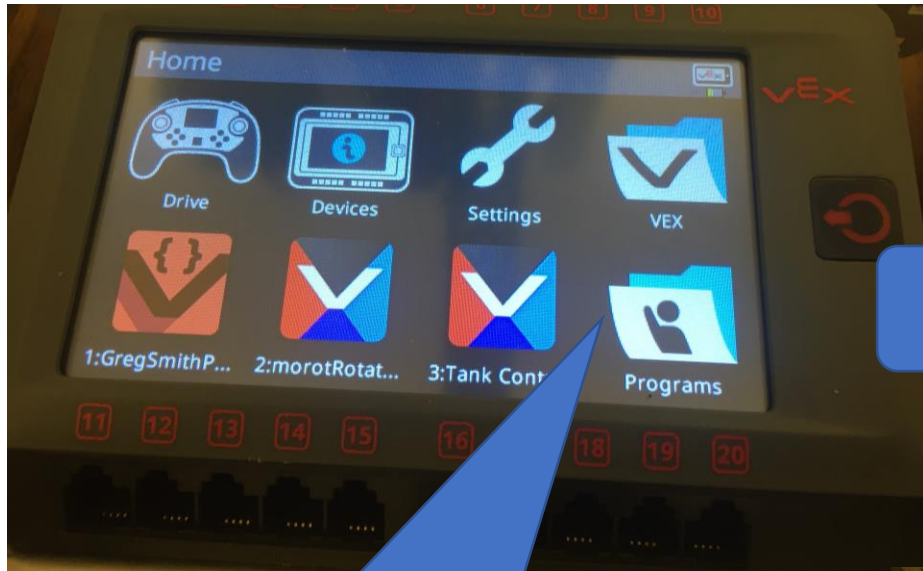
To Run:
Press the Run Triangle



To Stop. Press and hold the Power Button for a little. Not too long or it will turn off the Brain.



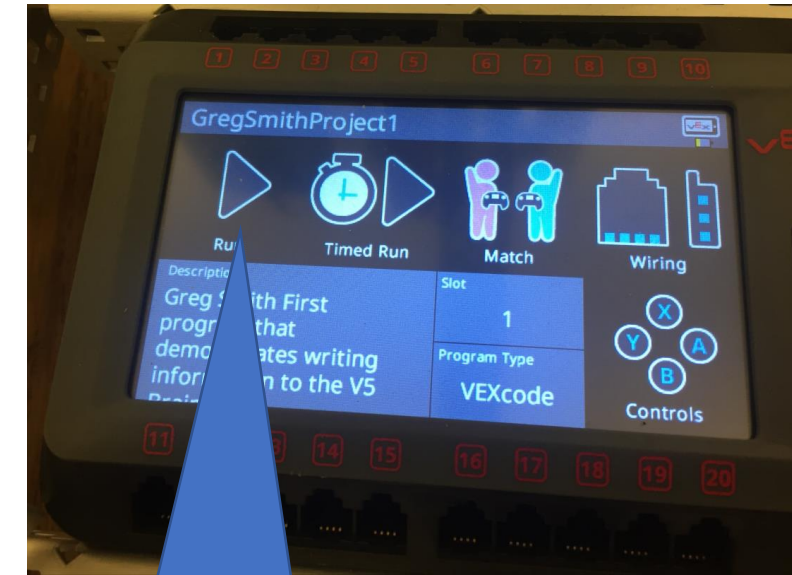
Running previously downloaded program on the V5.



1) Click on 'Programs' folder

Note: Press the Power Button to back out and get to the stop button..

2) Click on the Program you want to run



3) Click 'Run'



Coding Syntax Notes

1. Punctuation

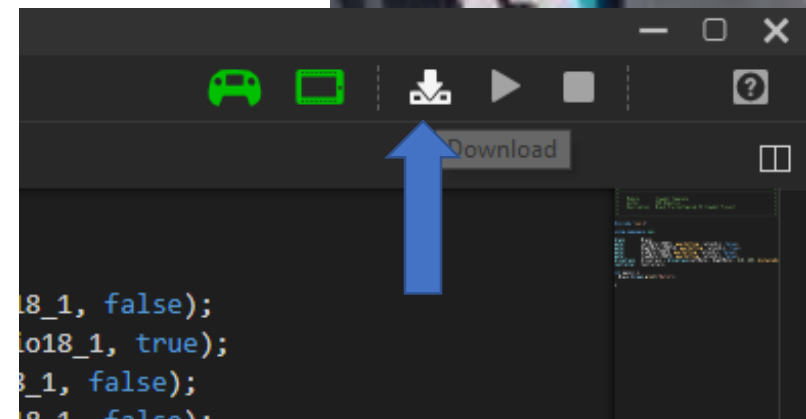
1. () Used for the arguments sent to the command
2. ; Marks the end of a command
3. "" Surrounds the Characters that will show up on the screen when using Brain.Screen.printAt()
4. // For a one line comment
5. /* */ Multi line comment
6. Brain.Screen.printAt(x, y, "String that shows on the screen.");
7. All commands AnD vArIaBlEs ArE CaSe-SeNsitivE!!!

Code, compile, download and run a program. You can use the code below or experiment with other Brain commands.

```
18
19  int main()
20  {
21  |   Brain.Screen.printAt( 10, 20, "The Dalles Loves February 2/15/2020!");
22  |   }
23
```

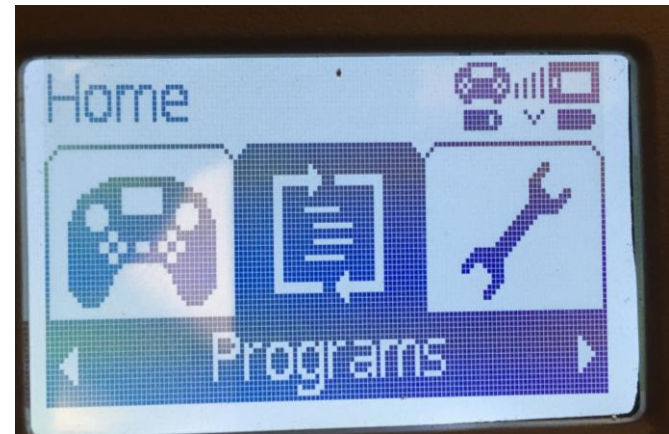
Downloading Wirelessly

- Pair Brain with Remote with VEX Smart Wire
 - Connect the brain and remote with a wire, turn them both on and look for the connected trailer in the top right of the brain.
- Connect Remote and Brain Wirelessly
 - Remove the wire that connects brain and remote
 - Wait for connection confirmation (Top right of brain)
- Connect Remote and computer with USB Cable
- Download the program
 - If all went well VEXcode now uses the remote to download the program wirelessly!!



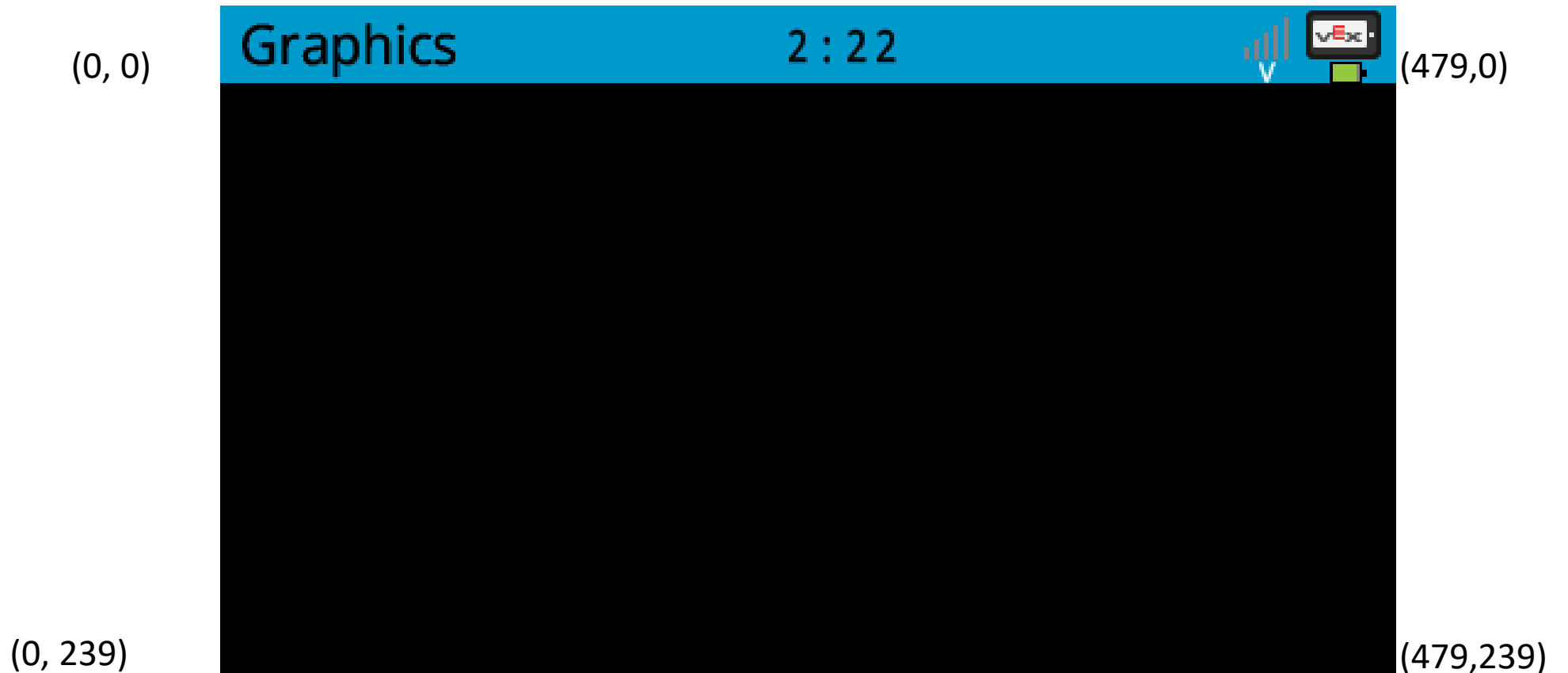
Running the program from the remote

- Use the left or right arrow keys to get to 'Programs'
- Hit 'A' to display the programs
- Use the left or right arrow keys to pick the program to run
- Hit 'A' to select the program
- Hit 'A' again when the play triangle is highlighted to run the program
- Use the computer to stop the program or
- Stop the program from the Brain
 - Hit the Power Button to move up a menu level
 - Hit the stop button



Adding Graphics on the Screen: Dimensions

- The Screen has input and output



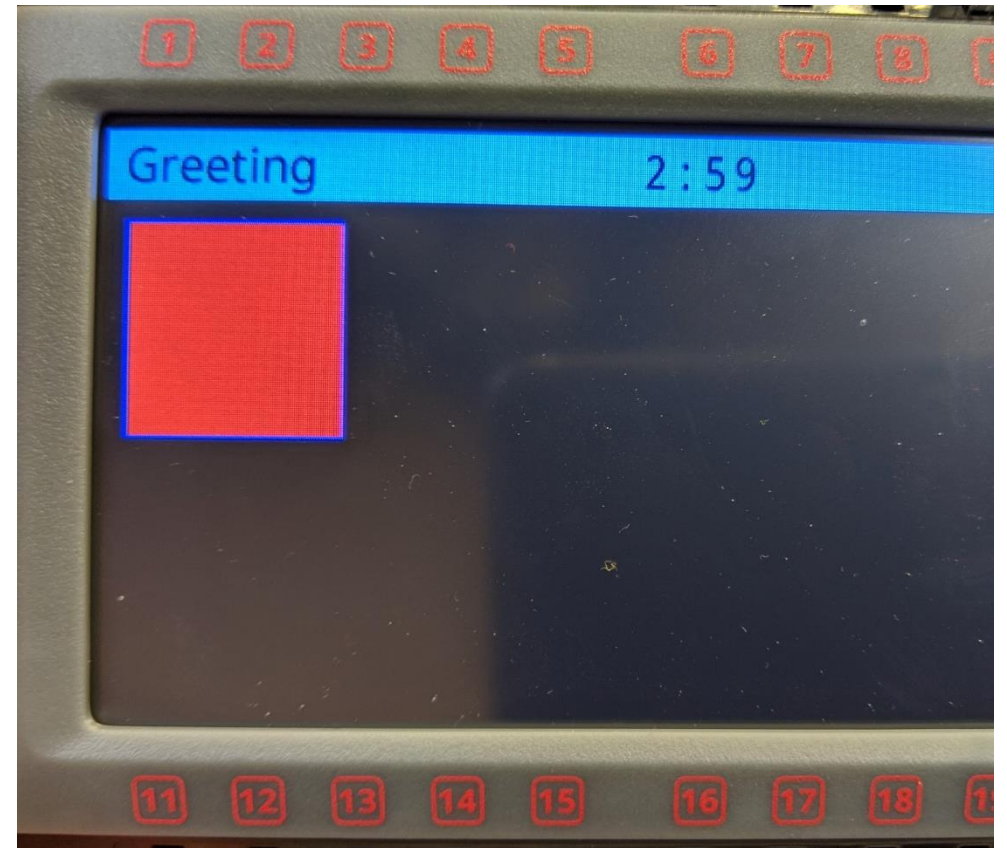
Programming Example: Putting Graphics on the Screen

```
4  int main() {  
5      Brain.Screen.setFillColor(color::red); ← Setting  
6      Brain.Screen.setPenColor(color::blue); ← Setting  
7      Brain.Screen.drawRectangle(10,10,100,100); ← Action  
8  }
```

X Y W H

Save, Download Run & Observe!

- Save your program (YourNameGraphics) , download it to the robot, and observe the behavior.
- Change the values in your program to observe how those changes affect the output.
 - Remember, any time you make changes to your program, you must download them to the robot for it to take effect.
- To stop the program execution, you can press the physical button on the Robot Brain, followed by the Stop button on the screen.



Resource: Using the Autofill to experiment Brain.Screen. ...

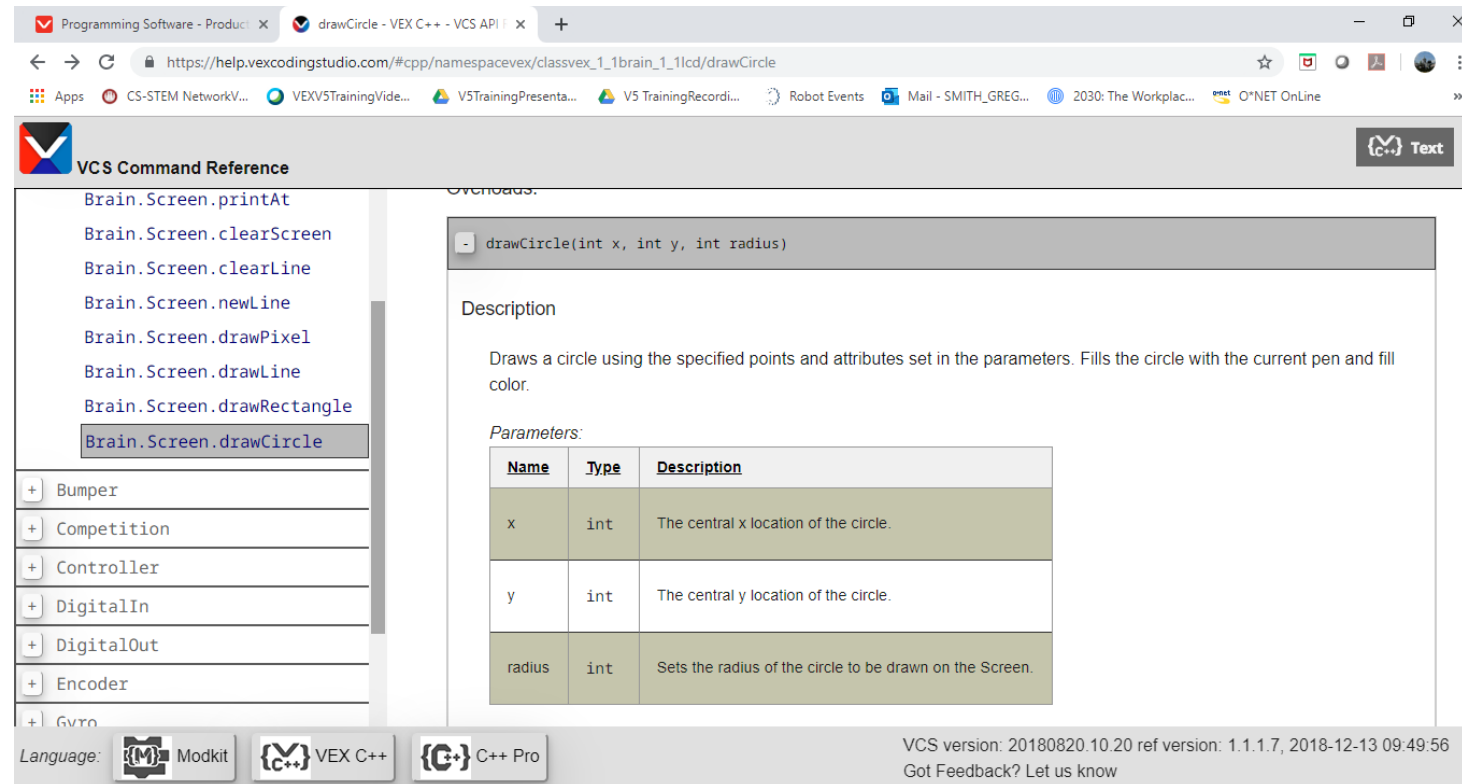
```
int main() {
  Brain.Screen.
  //Head
  Brain.Screen.
  Brain.Screen.
  //Eyes
  Brain.Screen.
  Brain.Screen.
  Brain.Screen.
  Brain.Screen.
  drawImageFromBuffer(uint32_t *buffer, int x, int y,
  drawImageFromBuffer(uint8_t *buffer, int x, int y,
  drawImageFromFile(const char *name, int x, int y)
  drawLine(int x1, int y1, int x2, int y2)
}

text    data    bss    dec    hex filename
6540    1088    1063760 1071388 10591c build/GregSmithFace.elf

[info]: build completed!
[info]: download
[info]: download completed!
```


Resources Online: Same commands as VEX Coding Studio.

- *Help.vexcodingstudio.com*
 - Vex Coding Studio Command Reference
 - Gives a description of how the commands work, often with sample code.



The screenshot shows a web browser displaying the VCS Command Reference page for the `drawCircle` command. The page title is "VCS Command Reference" and the URL is `https://help.vexcodingstudio.com/#cpp/namespacevex/classvex_1_1brain_1_1lcd/drawCircle`. The left sidebar lists various command categories, with `Brain.Screen.drawCircle` selected. The main content area shows the command signature `drawCircle(int x, int y, int radius)` and a description: "Draws a circle using the specified points and attributes set in the parameters. Fills the circle with the current pen and fill color." Below the description is a table of parameters:

Name	Type	Description
x	int	The central x location of the circle.
y	int	The central y location of the circle.
radius	int	Sets the radius of the circle to be drawn on the Screen.

At the bottom of the page, there is a footer with the text: "Language: Modkit VEX C++ C++ Pro" and "VCS version: 20180820.10.20 ref version: 1.1.1.7, 2018-12-13 09:49:56 Got Feedback? Let us know".

Getting input from the touch screen

```
int main() {  
  while (true) //While true is true, repeat the commands in the next {}  
  {  
    while (!Brain.Screen.pressing()) //While NOT (!) the screen is being pressed repeat the commands in the next {}  
    {}; //Do nothing. Will only get out of this loop when the screen is pressed.  
    //Say 'Ouch' where the screen was pressed  
    Brain.Screen.printAt(Brain.Screen.xPosition(),Brain.Screen.yPosition(),"Ouch");  
  } //Go back to while (true) to repeat forever  
}
```

