technocamps

Inspiring | Creative | Fun
Ysbrydoledig | Creadigol | Hwyl

# App Inventor meets NXT Workbook

i.t.wales
www.itwales.com
Swansea University
Prifysgol Abertawe
Prifysgol Aberystwyth University
Prifysgol Bangor University
University of South Wales
Prifysgol De Cymru
ESF
Llywodraeth Cymru
Welsh Government
Ewrop & Chymru: Buddsoddi yn eich dyfodol
Cronfa Gymdeithasol Ewrop
Europe & Wales: Investing in your future
European Social Fund

# App Inventor meets NXT

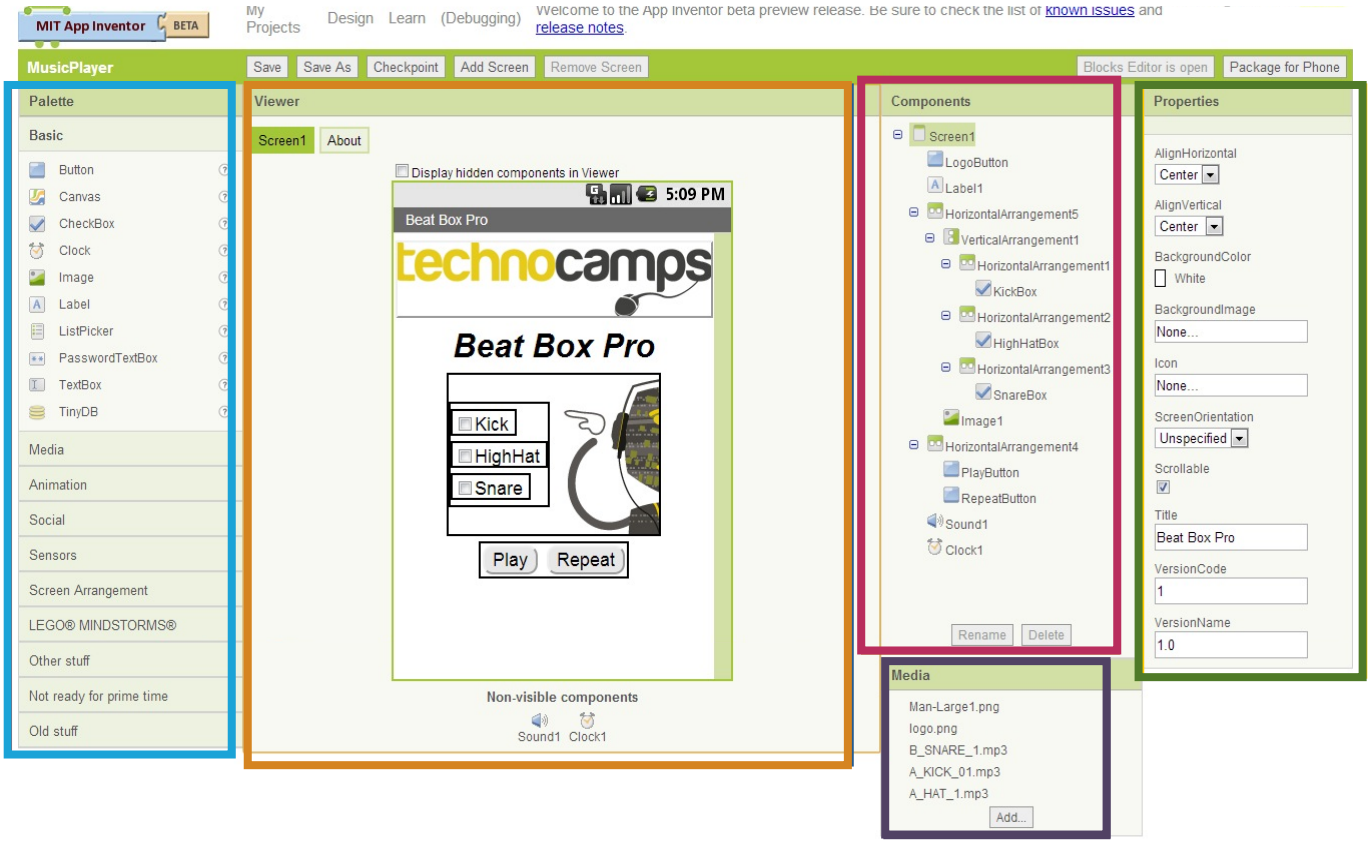
## Note:

**To begin, build your NXT robot using 2 motors, connecting each to the NXT. This will be controlled using the android application, to create the application follow the instructions within this workbook.**

App Inventor is a cloud-based application development tool, enabling users to develop Android applications for free! App Inventor has recently updated the software, however many schools have the initial version installed on their systems, so this workbook will be looking at the first version of the tool.

This can be accessed via this link: http://beta.appinventor.mit.edu/

To begin, select "Invent" and enter your gmail account to log in with a username and password. Once you have successfully logged in, you should be taken to a screen that displays a variety of projects you may have developed in the past. All projects created will be stored here, even after downloading onto an android device. Open a new project and the screen similar to below will appear, without the components already added:




See the colour coding below for more information:

**In this box are all the components you can add to your application. You just simply click on which one you want and drag the component onto the "Viewer" panel.**

# App Inventor meets NXT

**This panel displays what your application will look like when it first starts. It shows you visually, each component that you have added to your application.**

**This box contains a list of all the components that are currently in your application. You can rename and delete them in this panel.**

**This panel contains all the "Media" (images and sounds) which live in your app. Any media that you wish to use must be added in this panel.**

**These are the details for the components you have selected in the "Components" panel. Here is where you set the initial behavior of the component.**

Now that you are familiar with the App Inventor designer window interface, you will be using some of the components listed on the left hand side of the screen to begin building your application.

## 1) Components

Add 2 buttons to your application. These need to be called "ConnectButton" and the other "DisconnectButton". The names cannot have spaces and the use of capital letters. These will be given code using the Blocks Editor a little later on in this workbook.

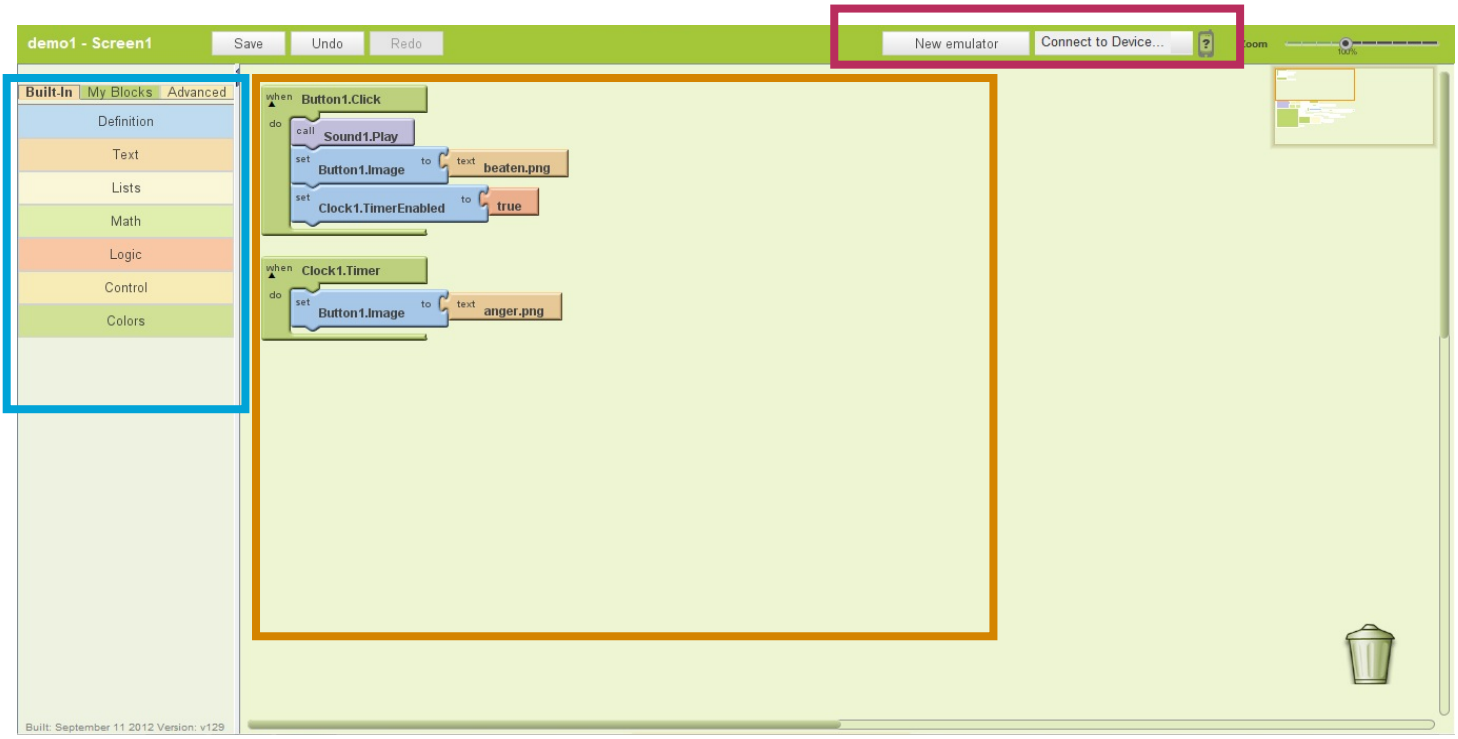
Add each of the following components. These are invisible so cannot be seen on your app, however they will be given code later on that will effect the functionality of the application:

- Bluetooth Client
- Drive Clock
- NxtDrive Motor (add 2 of these, 1 for each motor)

These may not need renaming, however if later on you feel it would make it easier for your coding to rename them, then you can do so using the column to the right of the application.

Next you will be using the Blocks Editor to begin developing the code for your application, telling the components what to do, how to do it and giving the application functionality. On the following page is a breakdown of the Blocks Editor interface. To open the Blocks Editor, click on the "Open Blocks Editor" button in the top right hand corner of the screen. You may be required to update Java on your computer to ensure the blocks editor runs properly. When it opens, you should be able to see the image at the top of the following page.

# App Inventor meets NXT




This panel has three choices at the top: “Built-in”, “My Blocks” and “Advanced”. The Built-in panel has a list of built in blocks that you can use. The my blocks panel has a list of all of the components in your app and if you click on those it shows you the blocks for each. Lastly, the advanced panel contains some advanced blocks.

This panel contains all the current blocks you have in your application that are enabling it to do something. You can add more and more blocks and scroll around the page to view and change your current blocks.

You can connect to your device by clicking the drop down box and selecting your device. Or you could run an “Emulator” which is an artificial device.

Now that you are familiar with the blocks editor, try dragging on combinations of code to see how they connect together. Build a familiarity with the different categories described above, noting which segments of code require more instructions added to them. Try clicking anywhere on the screen and note the options bar that appears, enabling quick, and easy access to some of the code from the menu on the left.

Before you start the next task make sure you drop all existing code into the bin on the right hand side to clear the development environment ready to begin.

## 2) ConnectButton.BeforePicking

Make sure you only have 1 screen for this workbook and that it has been selected on the Designer window This will ensure when the Blocks Editor is used, that the code is applied to the correct screen.

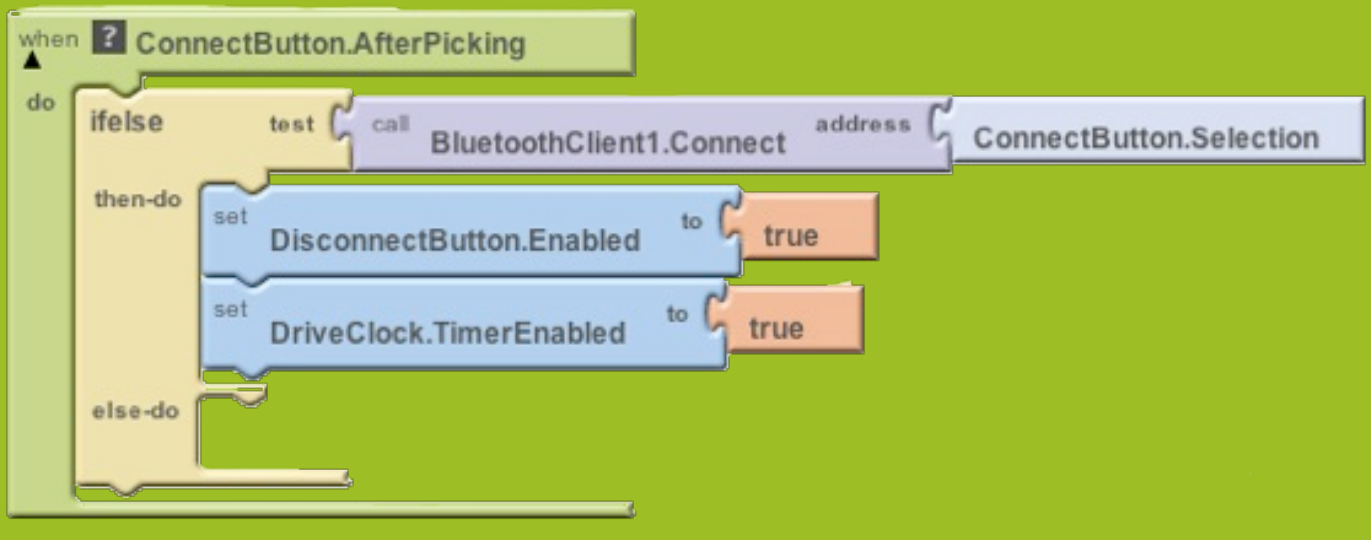
Add the following code to the blocks editor, building up the code slowly using the colour co-ordination as a guidance as to where to locate the different segments of code in the menu on the left. For instance, if the code is referencing a component on your Designer screen, click on "My Blocks". To begin, add the following:


You can comment on each of these blocks of code, to do so right click on the code and select "Add Comment". A comment is used to describe the code (particularly helpful if the code is complicated and not very user-friendly). Type the following as a comment for the "ConnectButton.BeforePicking" code:

```
"When ConnectListPicker is pressed, show a list of the robots that have been
                        paired with the phone."
```

## 3) ConnectButton.AfterPicking

You will be adding an event that describes what occurs after the ConnectButton has been selected. Add the following:



### 4) ConnectButton.AfterPicking continued

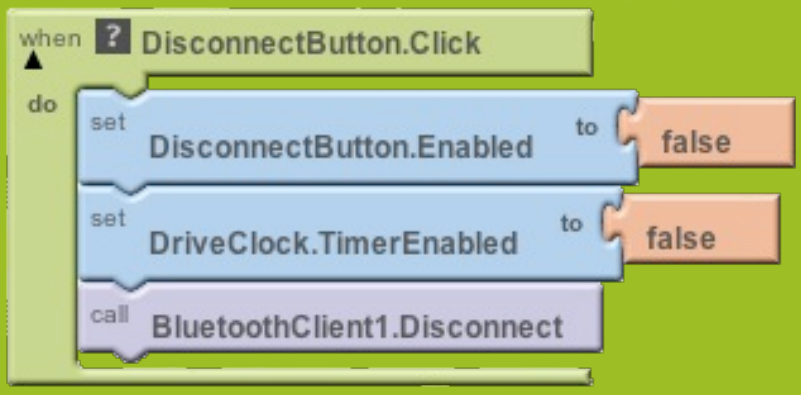
You can see from the previous activity, that the code is incomplete, the IF ELSE statement required more code added to the ELSE element. You need your code to output an error message. How can you do this? Try and change the text of the connect button to "Error" to alert the user of an error with the connection. The segment of code below can be used, this can be found in the "My Blocks" category, however the below code needs completing.


Alternatively, you could add a label to your Designer screen that is invisible, but appears with an error message as a result of the "ELSE" element of this code, depending how confident you are with App Inventor. You can comment on each of these blocks of code also.:

```
"After a robot has been chosen, connect to the robot.
If successful:
      - Hide ConnectListPicker.
      - Show RobotControlPanel.
      - Show Disconnect Button.
      - Turn the robot's light sensor on.
Otherwise:
       - Show an error message."
```

### 5) DisconnectButton.Click event

This button will disconnect the bluetooth connection between the app and the NXT. Add the following code to the Blocks Editor and apply the comment on the right of the code to assist:

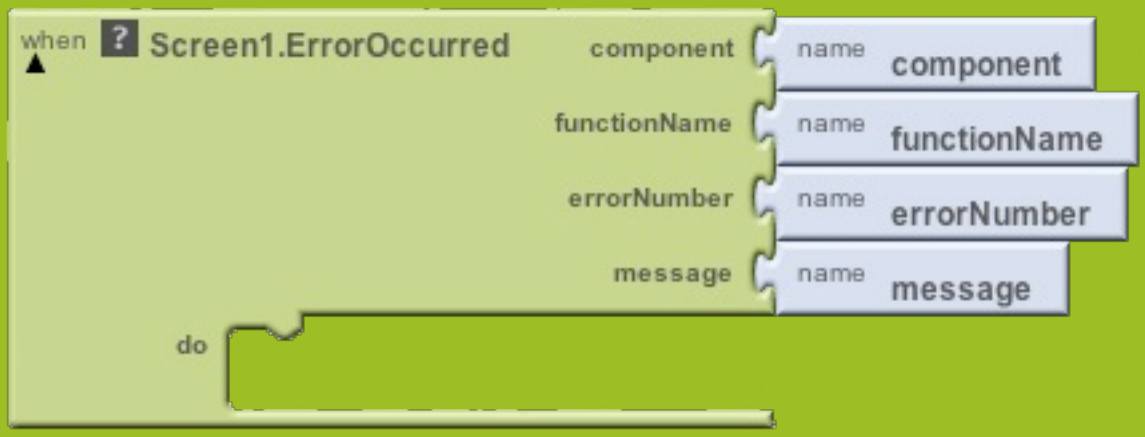



```
"When DisconnectButton is pressed:
- Turn off the robot's light sensor.
- Close Bluetooth connection
- Hide RobotControlPanel.
- Hide DisconnectButton.
- Show ConnectListPicker."
```

# App Inventor meets NXT

## 6) ErrorOccurred

This will output an error message as a result of any errors occur. Add the following code to the Blocks Editor:



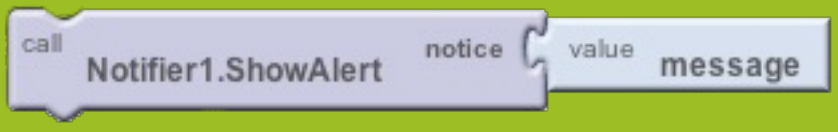

Next you need to give this code some comments, reminding yourself and other users of how it works:

```
"If an error occurs, show the error message on the phone's screen."
```

## 7) ErrorOccurred continued

Have you noticed that the "Screen1.ErrorOccured" When command is incomplete?

Add a "Notifier" onto your Designer window and go back to the Blocks Editor. To complete the "When Do" statement, add the following code for "Do" to alert the user of a message. This will output an alert with an error message to the user of the application:




You now have the basics for your application to continue building upon. You will need to return to the Designer window of App Inventor in order to add more components to your application. At this stage you could also focus on the appearance of your application, having a theme and improving the application visually.

Don't delete any of these original components as this will cause problems for the associated code in the Blocks Editor.

# App Inventor meets NXT

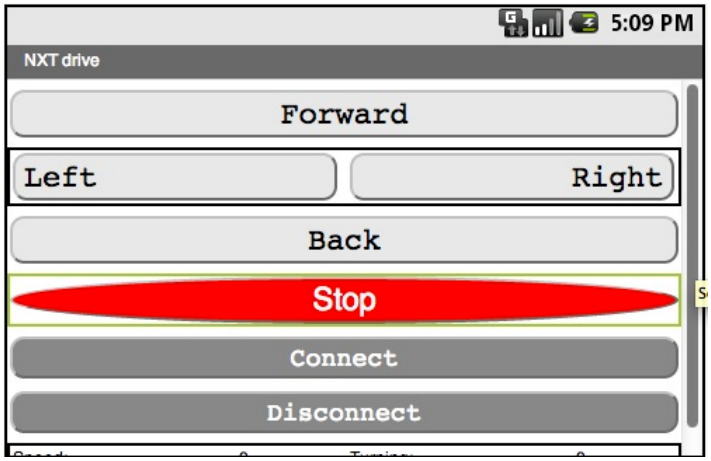
## 8) Additional components

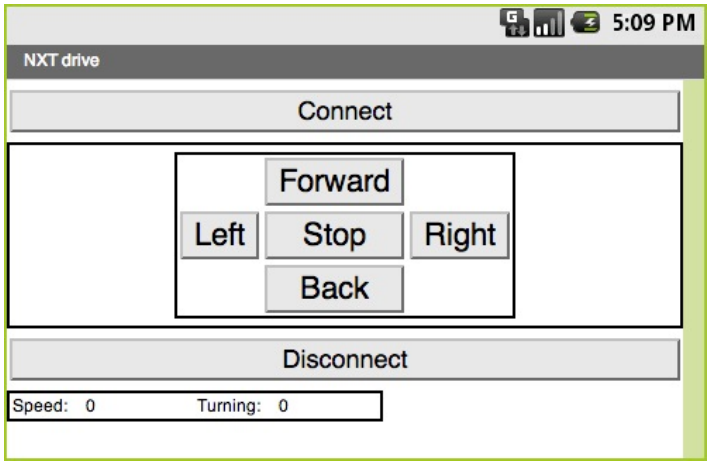
Add the following components to your screen:

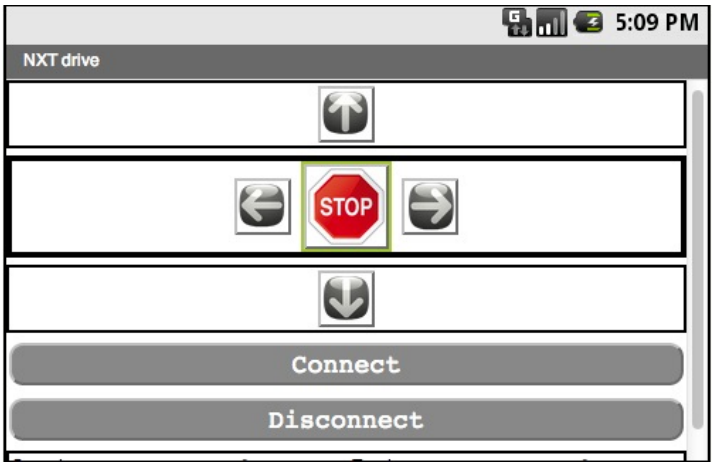
- Button labelled "Forward"
- Button labelled "Backward"
- Button labelled "Right"
- Button labelled "Left"
- Button labelled "Stop"
- Label with the text "Speed"
- Label with the text "Turning"

At this stage you can also edit the appearance of these components to suit your desired theme/appearance. You can do this by adding screen arrangements and editing each of the component's properties using the column on the right hand side of the Designer window.

Some examples of how to structure your application are below:



# App Inventor meets NXT

Next you will need to return to the Blocks Editor, here you have added code for "ConnectButton", "DisconnectButton" and errors alerts. Now that new components have been added, you need to begin building the rest of your code.

### 9) Add variables

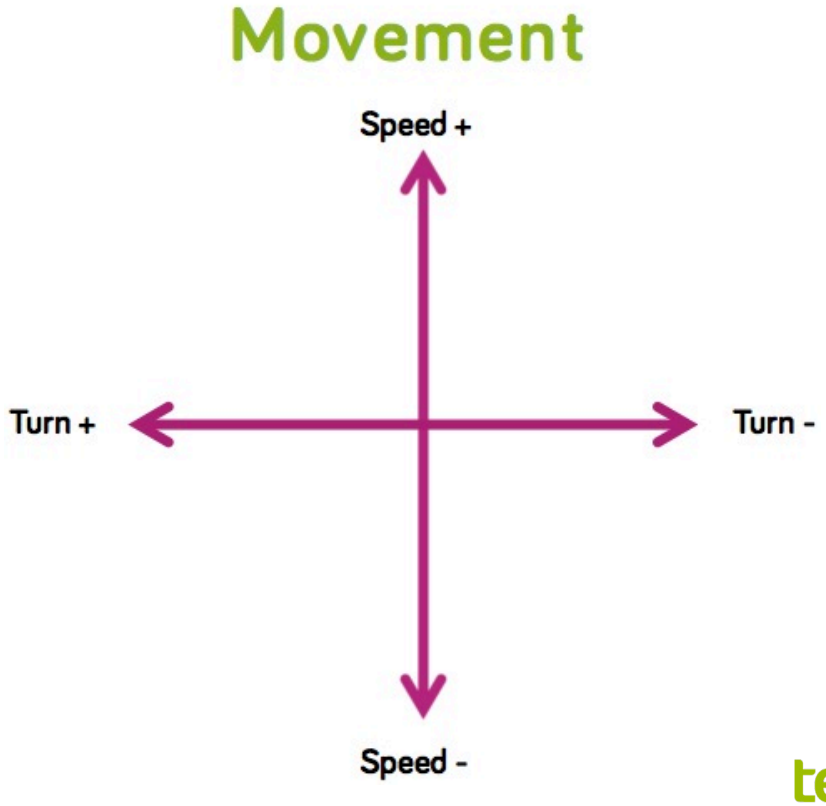
You have already added 2 labels to your screen, so the next task is to create variables, which calculations can be performed on to change its value. These values can then be output onto labels e.g. score = 10 points.

A variable is a part of memory that is saved away inside the application. This chunk of memory can be named anything, for instance "score". The information stored inside a variable can be of any format, for example it could be text or a number. This information can be called upon later on, and can be used in some manner e.g. Score - 10 points because an enemy in a game shot at the main character.

Add 3 variables called "Speed", "minTurn" and "Turn", these will be set to take numbers. Set each value to 0 for now, these can be edited later on.

For the robot to move correctly you need to begin adding functionality to the directional buttons on the application. Here you will begin applying the variables also.

Below is a diagram of how the robot will move depending on the manner in which the variables are implemented into the code for the buttons. To begin, have a look at how to move forward.



# App Inventor meets NXT

To move straight forward, you do not need to use "Turn". Currently the "Speed" variable is set to 0, which will not move the robot. So to progress forward you need to add a number to the current value of speed, for example:

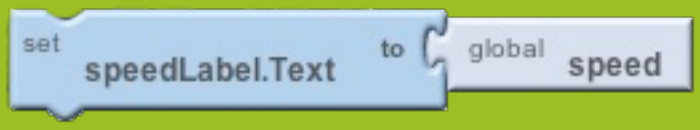
*Set the "Speed" variable to* the current *speed* + *10*

### 10) ForwardButton

Firstly, add the "ForwardButton.Click" event to begin giving your forward button some functionality.

Next add the calculation "Set speed to (Speed + 10)" using the "My Blocks" menu and the "Math" category.

To output the speed to the app user, you can update the "speed" label on the Designer window to contain the new value of the speed using the following code.




### 11) BackwardButton

How do you think you could move the robot backwards?

In the previous activity you have added values onto the "speed" variable in order to move the robot forward. See if you can implement this yourself, similar to the calculation above but getting the value of speed to decrease rather than increase.

Don't forget to change the speed label to the updated value of the speed variable from your calculation.

To move left and right you need to begin using the "turn" variable rather than just speed. Refer back to the diagram at the bottom of the previous page to do the next activity.

### 12) LeftButton

See if you can implement the calculation, similar to the previous 2 activities but using the "turn variable" for left. Don't forget to change the turn label to the updated value of the turn variable from your calculation.

# App Inventor meets NXT

### 13) RightButton

See if you can implement the calculation, similar to the previous 3 activities but using the "turn variable" for right button. Don't forget to change the turn label to the updated value of the turn variable from your calculation.

To give your stop button functionality you need to ensure that there is no speed and no turn value either. To stop your robot, both variables for speed and turn need their values changed to 0, and for user feedback purposes the labels would also require updating on the application informing users of this change in speed and turn values..


### 14) StopButton

So far you have experience of setting existing variable values to another value, for the stop button you require your speed variable and turn variable to be returning to 0. You can also update the speed and turning labels also to offer a user more feedback.

You have already added a "Clock" component onto your application but what does it actually do? A clock allows us to keep changing how we drive. You can use a clock to keep adjusting the motors to the values of speed and turn, using the "timer" event in the Blocks Editor.

But what values do you give your motors? So far you have established that if you are not moving (stopped) that speed = 0 and turn = 0 (also the values of your variables). If you are not moving, you need both of your motors to stop.


You have already used conditional statements, you can use them again here to test that if "speed = 0 and turn = 0" then both motors need to stop.

### 15) DriveClock

Add the "DriveClock.Timer" event to the Blocks Editor. You need to perform an IF ELSE conditional statement, testing if both speed and turn equal 0. Add this onto the "DriveClock.Timer". For the test element of the conditional statement, it requires 2 tests that need to both be true to complete the command. To add these tests, go to the "logic" category in the menu and use the "and" command to state that both the speed variable and the turn variable need to equal 0.

You need to add for the "then" element of the conditional statement to stop each motor, these can be found in "My Blocks" under each of your NxtDrive components. Don't' forget to do this for both of your NXT Motor Drivers.

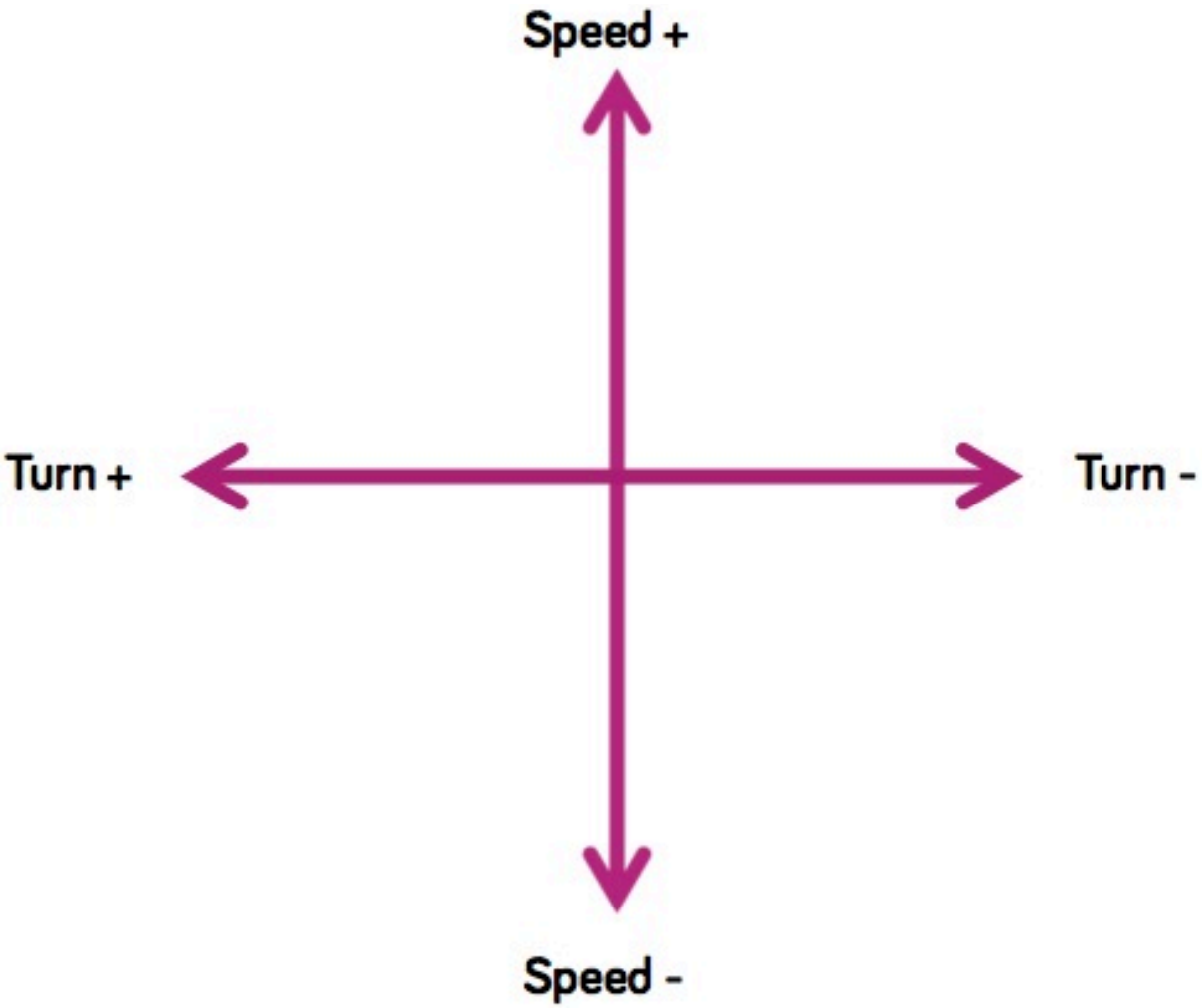
There is still a part of the IF ELSE statement incomplete, requiring you to state what instructions will be given if both speed and turn don't equal 0. As you are aware from the code already entered, to move your robot forward, the speed needs to be positive.

Next you need to consider the "ELSE" part of the conditional statement. Here you will be applying the speed calculations to the motors that move the robot around. As you are aware there are 2 motors, left and right so the next element of code needs to state the following:

```
If speed > 0
     Then
          Move left motor forward
          Move right motor forward
```

For testing and example purposes, consider the following values for your robot movement calculations:

Speed = 30
Turn = 20
MinSpeed = 50
Left Motor = 0 Right Motor = 0




As you can see from the diagram above, the turn to the left is a positive value and to the right is a negative value. Therefore with a turn of "20" the robot would be turning to the left.

### 16) DriveClock IF ELSE conditional

Using the above information as a guide, implement the following pseudocode as an embedded IF ELSE statement under the "DriveClock.Timer" ELSE.

```
If (Speed > 0)
Then
     RightMotorDrive.MoveForwardIndefinitely (Speed + Turn + MinSpeed)
     LeftMotorDrive.MoveForwardIndefinitely (Speed - Turn + MinSpeed)
Else
     RightMotorDrive.MoveBackwardsIndefinitely (0 - Speed) - Turn + MinSpeed
     LeftMotorDrive.MoveBackwardsIndefinitely (0 - Speed) + Turn + MinSpeed
```

### 17) ConnectButton.AfterPicking edits

You have previously created an event within the Blocks Editor called "ConnectButton.AfterPicking", which enabled the disconnect button and DriveClock.Timer. Add the following "pseudocode" enabling the rest of the buttons after the application has connected to the NXT via Bluetooth:

```
If ...

Then:

  - Enable stop button (TRUE)
  - Enable left button (TRUE)
  - Enable right button (TRUE)
  - Enable forward button (TRUE)
  - Enable back button (TRUE)

Else ...
```

Don't forget to apply the boolean values of true and false, stating that "It is true that button x has been enabled".

### 18) DisconnectButton edits

You have previously created an event within the Blocks Editor called "DisconnectButton.Click" which currently disconnects the bluetooth connection to the NXT from the application, disables the disconnect button as well as disabling the "DriveClock.Timer".

In addition to these, add the following pseudocode to this event:

```
  - Enable stop button is FALSE
  - Enable left button is FALSE
  - Enable right button is FALSE
  - Enable forward button is FALSE
  - Enable back button is FALSE
```

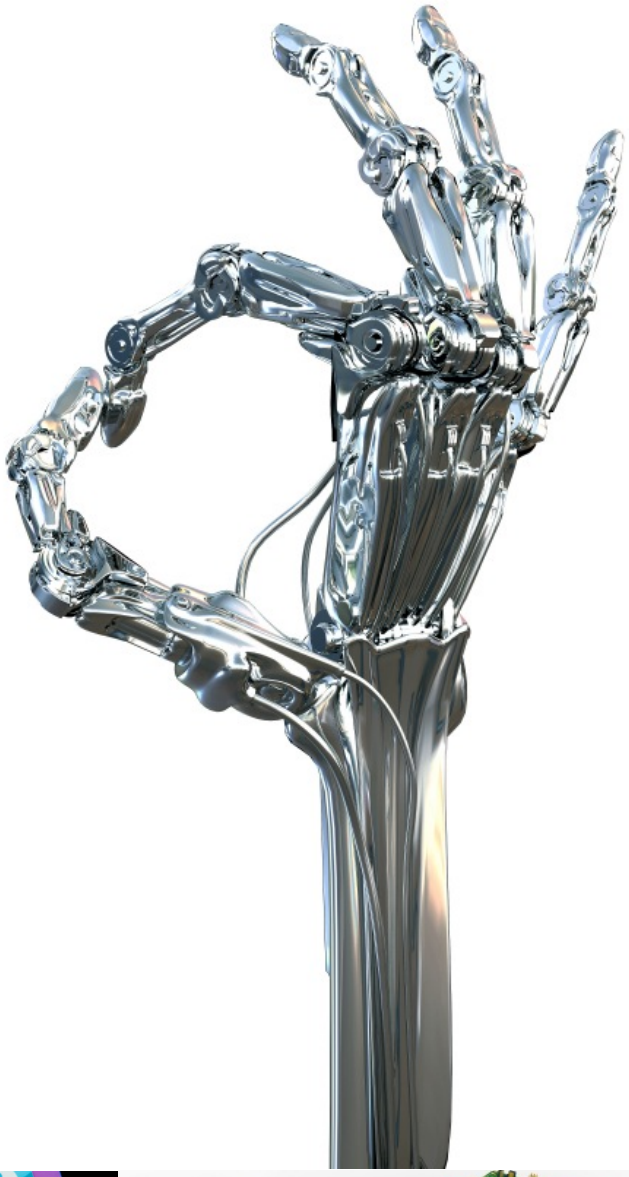
Don't forget to apply the boolean values of true and false, stating that "It is false that button x has been enabled".

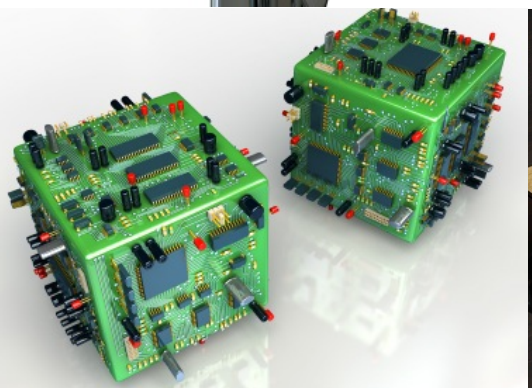
### 19) Testing

You can upload the application for free onto your computer or connect your Android device using the appropriate USB connection. Testing is an important element of app development and software development on a whole, sussing out any small mistakes in the code and deciding upon how the application can be improved both visually and functionality wise.

Here are some factors to consider when testing your robot:

- What would you improve about your application?
- Does it connect to the NXT okay?
- What is the user feedback like?
- What do you think you would change about its appearance now that you have seen it properly on your device?



# technocamps

www.technocamps.com

@Technocamps

Find us on Facebook