

# Tutorial 2

**Disclaimer:** In your life as a software engineer, it will be important to know how to use third party software platforms and their materials. As such, **many of the materials we recommend for this course will be third party.** This will differ from some of your other programming courses, where you will need to write most of the software yourself. It is important to develop both sets of skills, that is, using third party materials as well as knowing how to write your own programs.

Now that you're becoming more familiar with MIT app inventor, it's a good idea to spend some time learning some of the different android features you can use with this tool. In this tutorial sheet, you will find some notes and quick-links to assist with learning different features.

During the next two weeks, we'll look at the android phone sensors and some other miscellaneous features which can be used with MIT app inventor. These features should help you with your project later down the track.

So that you can begin designing your project in week 4, there will be many features presented over the next two weeks. But remember, **many hands make light work** 😊. Now might be a good time to talk amongst your group members, and assign learning/practicing different features to different groups members, so that as a team, you have good skills coverage.

## Sensors



This week, we'll provide you with some information on all of the sensors the MIT app inventor platform provides support for. These are:

- accelerometer
- barcode scanner
- clock
- gyroscope sensor
- location sensor
- near field communication
- orientation sensor
- pedometer
- proximity sensor

To help familiarize yourself with the tool, we will include links to existing tutorials for all sensors.

Then, we're going to suggest you flex your design muscles. For each sensor, we will propose a small request you can try to complete yourself. This might be a simple app, or a request for changes to an existing app from a tutorial.

**Sample solutions for these requests will be available later this week**, and it will be good to view your solution in comparison to the sample. Some of these requests will be classified based on the technique we are trying to teach:

	Tasks related to data collection or data maintenance can be identified with Yoshi.
	Tasks related to conditional statements and user interactions can be identified with a mushroom.

*Some sensors or sections in this tutorial are presented in grey. While you are free to learn as much as you like about these features (if they are of interest to you), they have been marked in grey because we feel they are not important features to learn in the context of this course.*

## Third Party Documentation

Before we begin on the sensors, it's a good idea to have a look at the third party documentation. In software engineering, when you are using a platform developed by another entity, the software will most likely be supported by documentation with details about the functionality available for the platform. The documentation for the MIT app inventor sensors can be found [here](#).

When reading the documentation on this site, you should pay attention to:

### Properties:

These are the member variables of the class (or sensor) you are working with. They are basically bits of information available with the sensor, which you can use to determine, or set the state of the sensor. For example, the xAccel, yAccel and zAccel are all properties of the accelerometer. You can use these properties to determine the acceleration the phone is detecting in the x, y and z axis at any time.

### Methods:

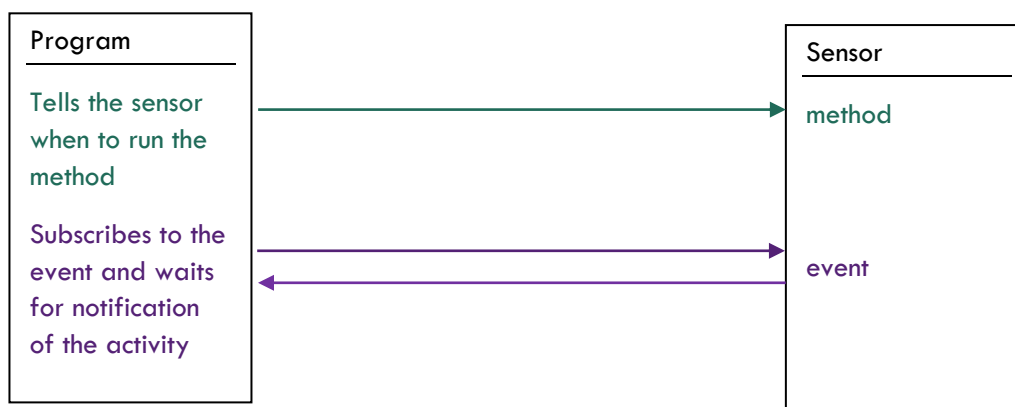
Methods allow you to give an instruction to the sensor to complete some task. For example, the click of a button might force a call to a method, such as DoScan for the QR scanner.

### Events:

Events are activities or actions which the sensor can detect. You can add code blocks to an event, so that when an event occurs, you can force something to happen. For example, the accelerometer has an event which can detect if the phone is being shaken. You can make use of this event, the instruct the application what to do when it notices this event occurring.

### Methods vs Events

In the most basic terms, a **method** is used to perform an action, such as updating some data, or requesting a specific activity. As a programmer, you will have the ability to decide when you wish to call a method. An **event** runs in response to an activity or set of conditions. As a programmer, you have the ability to instruct the program which events you are interested in. When that event occurs, your program is notified.



## Give it to me straight?

This week, we want you and your group members to familiarize yourself with the sensors available for android application development.

There is a lot going on here, so remember, **many hands make light work!** It could be a good idea to discuss with your group the different sensors, and distribute the tutorials and exercises amongst yourself.

## Accelerometer

The accelerometer is one of the most important features of the smartphone. It provides information about the movement of the phone, letting us know the speed and direction of travel in 3 dimensions (x, y, z). The measurements for x, y are on a scale of  $[-10, 10]$ . While the z axis gives a measurement of how gravity is affecting it. When stable, the z axis should give a reading of around  $\sim 9.8$ . In this case, measurements higher than this would suggest your phone is falling toward the ground, while measurements lower would suggest your phone is being raised/lifted away from the ground.

Note that the reading of the z axis may differ from phone to phone. Some phones will display a positive reading of  $\sim 9.8$  for a phone sitting screen side up, while other phones will display the inverse (a negative reading  $\sim (-)9.8$  for a phone sitting screen side up). The z axis is dependent on how your phone OS has been calibrated.

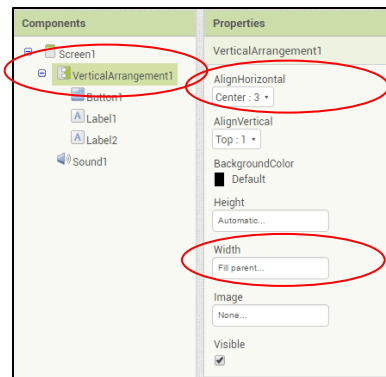
Rapid changes in the accelerometer values are used to detect shaking and phone movement. An MIT Tutorial using this accelerometer shaking component can be found [here](#). Note that this is an excellent beginner's tutorial for the platform. **It is recommended that a member of your group complete this tutorial.**

- Once you have completed this tutorial, we suggest you make the layout on the screen a bit better.

In the **designer** page, change the **VerticalArrangement1** component settings:

Set AlignHorizontal to: Center : 3

Set the width to: Fill parent



### Have a go yourself!



Build an application which displays pure recordings from the x, y and z axis dimensions. The application should have a controller which will allow the recordings to begin and end.

**Sample solutions will be provided next week.**

### For students using the emulator

Note that this feature is not available for students using the emulator. This sensor will work for students using an android phone via a wifi connection or USB connection.

## Barcode Scanner

The barcode scanner can be used for scanning both regular barcodes or QR codes. Bar codes are typically found on consumer product packaging while QR codes can be used to embed any string of text.

Note that at this time, there is no MIT tutorial on this feature. However, another detailed tutorial for this feature can be found [here](#). This is an excellent tutorial, that will scan either a barcode or QR code. It will then define what has been scanned, and in the case of a barcode, it will take you to information about the product. In the case of the QR code containing a web address, it will take you directly to the page **It is recommended that a member of your group complete this tutorial.**

When working with the Barcode Scanner, there are a couple of things you should also be aware of:

### Generating a QR code

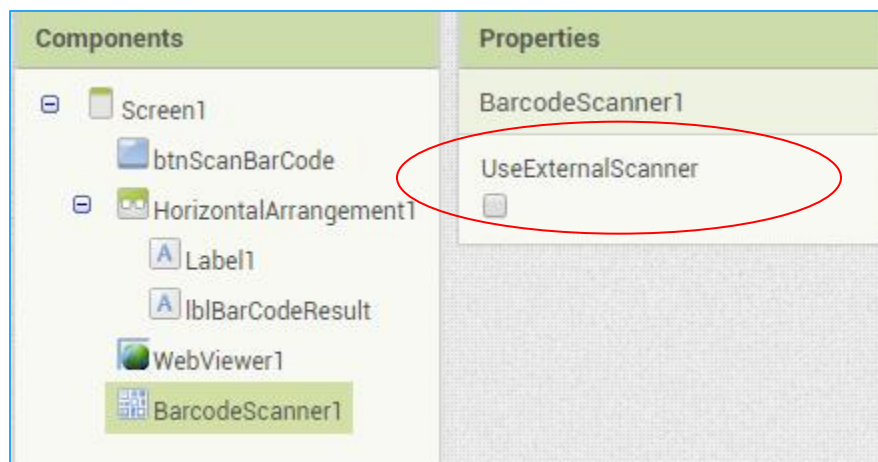
You can generate QR codes from [this](#) site.

1. Select Plain Text
2. Type in the text you wish to embed into the QR code
3. Select the colour
4. Download your QR code and save the file for scanning

### Other Notes

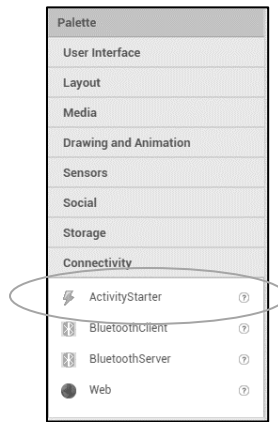
When using this feature, you have the option to use MIT app inventor's barcode scanner, or an external barcode scanner.

**To use MIT app inventor's scanner;** on the design page, select the BarcodeScanner1 component and ensure that the property, **useExternalScanner** is **unticked**.



**To use an external Scanner;** you will first need to ensure that the activity starter is included. The activity starter is a component which will let you fire up another application from within the application you are building. So by including the activity starter, you will be able to open up a barcode scanning application from within your application.

The activity starter can be found on the designer page, under Connectivity (in the Palette)



To add this to your app, simply drag and drop it onto the screen which is using the barcode scanner.

Your device will also require a barcode scanner application from the PlayStore. We can recommend **this** application.

Note that when using this feature, a common Error is: "Error 1501 - your device does not have a scanning application installed". If you receive this error, please ensure you have disabled the use of an externalScanner OR have installed a barcode scanning application.

### Limitations

The barcode scanner will only work with images viewed by the camera. It cannot scan the code of an image file which is stored on your device.

### Have a go yourself!



In this exercises, we would like to introduce using conditional statements to provide feedback to a user based on expected inputs/outputs

Create a QR code image, and keep a record of the text which is embedded.

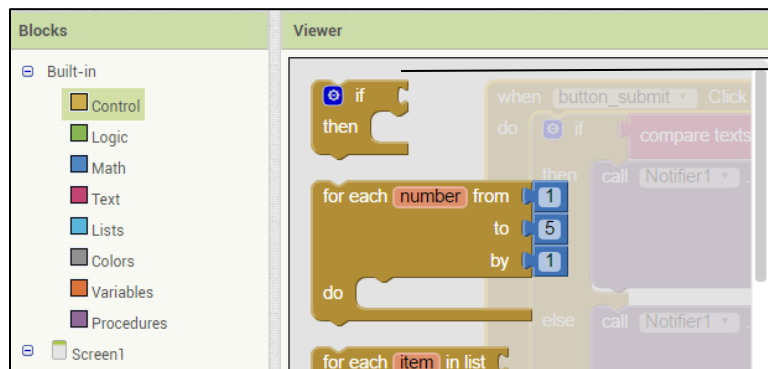
Set this text as a global variable in your App.

Use the barcode scanner to scan the image and check if the QR code text you are expecting match

If so, display a message informing the user of the successful identification.

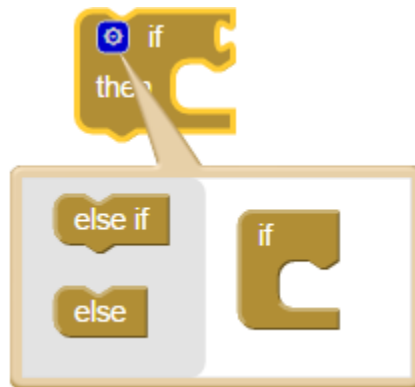
If not, display a message informing the user of the unsuccessful identification.

*Hint: for this exercise. You will need to use the “If” control block. This will be found in the built-in components of the blocks editor:*



Use this guy for your conditional blocks!  
Simply drag and drop it into your solution

Once you have added the block to your editor, you can add the extra “else” or “else if” statements by clicking on the blue settings wheel – then drag and drop the statement you wish to add to the control block.



*Hint: for proper testing of this application, you should test how the application behaves in both instances. So you will need to generate another QR image which contains unexpected text.*

### *For students using the emulator*

Note that this feature is not available for students using the emulator. This sensor will work for students using a phone via a wifi connection or USB connection.

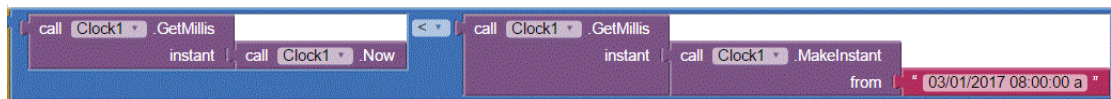
## Clock

All android phones come with an internal clock. The clock library offered by MIT app inventor allows you to get and set instances in time or make use of its timer component.

MIT Tutorials using the timer or instance in time component can be found [here](#): For experience with the clock timer, **we recommend the MoleMash tutorial**. In this tutorial, a picture of a mole will randomly jump around the screen. The player needs to tap the mole, to score points.

For experience with logging time instances, you can view the Pizza Party Tutorial. We do not recommend this tutorial, due to its limited use of clock features.

Note: The MIT tutorials do not cover an example of comparing an instance of time instance against another. Students may wish to note that the following sample of code allows you to make this comparison.



**How does this work?** All clocks in UNIX systems are set from the initial UTC time of 00:00:00 - 01/01/1970. The MIT app inventor allows you to convert any instance in time to the number of milliseconds which have passed since this initial time. The number of seconds which have passed will be represented as an integer. From here, you can run an integer comparison (using the math library comparison function) to find the relation between any two instances in time.

*Have a go yourself!*

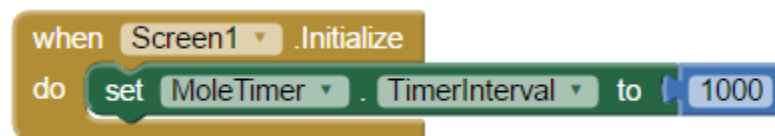
### Update the MoleMash Tutorial

Next to the reset button, include 2 new buttons for “faster” and “slower”

*Hint: You may want to use a horizontal alignment for these.*

When Screen1 is initialized, set the timerInterval to 1 second (or 1000 milli-seconds)

*Hint: The initialize method is an event in Screen 1, and the timerInterval is a variable in the moleTimer. You will need to use an integer from the Math library to set the value of the variable.*



When the button faster is pushed, make the mole move twice as fast.

*Hint: set the value of the timerInterval to its current value / 2*

When the button slower is pushed, make the mole move twice as slow.

Make sure you update the reset button, so that the time is re-set to its initial value when this button is clicked.

*For students using the emulator*

Note that this feature is not available for students using the emulator. This sensor will work for students using a phone via a wifi connection or USB connection.



## Gyroscope Sensor

The Gyroscope detects rotation and can measure angular velocity in three dimensions in units of degrees per seconds. The hardware for a gyroscope sensor is considered more sensitive and precise than the accelerometer, so while all android phones will likely have an accelerometer, it is often only the better ones that will have a gyroscope.

**So what is the difference between the Gyroscope and Accelerometer?** The main difference is that the gyroscope can sense rotation, while the accelerometer senses linear acceleration. That is the gyroscope tells you how fast you are rotating.

Consider placing your phone flat on a lazy Susan, and then spinning it. The gyroscope would tell you how fast it is spinning in each axis, but the accelerometer should only be reporting that gravity is acting on the z axis.

Note that at this time, there is no MIT tutorial on this feature. There is also little supporting material which we are able to find on this sensor.

*Have a go yourself!*



Create an app that displays the XAngularVelocity , YAngularVelocity and ZAngularVelocity at any point in time. Note that this application is like the exercise suggested for the Accelerometer.

*For students using the emulator*

Note that this feature is not available for students using the emulator. This sensor will work for students using a phone via a wifi connection or USB connection, provided the phone has this sensor included.

## Location Sensor

Android phones come with a location sensor. Using this sensor, it is possible to find the longitude and latitude of a device's current location. This information can then be used in applications such as Google Maps.

MIT tutorials using the location sensor can be found [here](#): **It is recommended that a member of your group complete one of these exercises: “Android, where's my car?” or “Map It: Displaying Locations on a Google Map”.**

Note that the above tutorials are advanced. Please allow a couple of hours for their completion.

### Have a go yourself!



Create a small application that stores as global variables the longitude and latitude of a desired location.

Add a button on the screen to test if you are currently at the location.

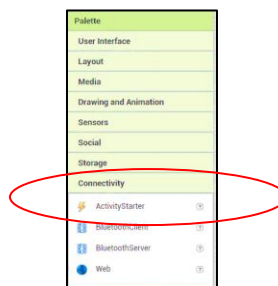
If you are at that location, provide a message to the user stating you are at this place.

If you are not, provide a message to the user stating that you are not detected at this place.

**Optional: Open up a Google Maps view pinned to the desired location (i.e; the global longitude and latitude)**

*Hint: for this you will need to use an Activity Starter.*

The activity starter is a component which will let you fire up another application from within the application you are building. So by including the activity starter, you will be able to open up a barcode scanning application from within your application. The activity starter can be found on the designer page, under Connectivity (in the Palette)



To add this to your app, simply drag and drop it onto the screen. Have a look at the **“Map It: Displaying Locations on a Google Map”** tutorial to see how locations are shown on a map using the activity starter.

Note that the optional feature is an advanced skill. Don't worry if you are unsure, a solution for this feature will be provided.

### For students using the emulator

Note that this feature is not available for students using the emulator. This sensor will work for students using a phone via a wifi connection or USB connection.

Note that the location sensor may not be able to detect the longitude or latitude coordinates from some areas of the university (for example, the tutorial dungeon).

## Near Field Communications Sensor

Near Field Communication (NFC) is a wireless radio communications standard (a little like wi-fi or Bluetooth) which is only applicable for short range communication. It enables two electronic devices, to establish communication by bringing them within 4cm of each other. The NFC readers are not particularly specialized, so NFC chips are incorporated into many smartphones.

An NFC chip in your phone works in conjunction with an NFC tag. The NFC tags are small, and don't have any power source. Instead, they literally draw power from the device that reads them, thanks to magnetic induction. When a reader (such as the chip in your device) gets close enough to a tag, it energizes it and transfers data from that tag.

It is an emerging technology, with increasing significance. It is embedded NFC tags incorporated into credit cards which provide the backbone for the development of tap and go payment systems, and is the driver behind mobile payment systems and mobile wallet solutions.

An MIT tutorial about this sensor can be found [here](#). **This tutorial is not recommended unless you have access to an NFC topaz 512 card, and a phone that can both read and write to NFC tags.**

*For students using the emulator*

Note that this feature is not available for students using the emulator. This sensor will work for students using a phone via a wifi connection or USB connection.

## Orientation Sensor

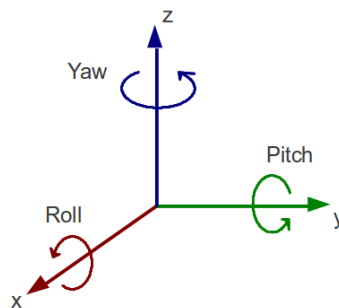
The orientation sensor is another component which can determine a phone's spatial orientation. That is, it will measure the phone's tilted angle in space. If you think this sounds like the accelerometer, then you are correct. In fact, it is now recommended by Google to use the accelerometer as the orientation sensor is its predecessor. **BUT IT IS STILL AN EXCELLENT SENSOR TO LEARN AND PLAY WITH** 😊

At this time, MIT tutorials for the use of this sensor do not exist.

However, because it can measure the degree of an angle, it is excellent for controlling the movement of an object on the screen. Sample tutorials for controlling the movement of an object can be found [here](#) and [here](#). **It is recommended that a member of your group complete one of these tutorials.**

### Have a go yourself

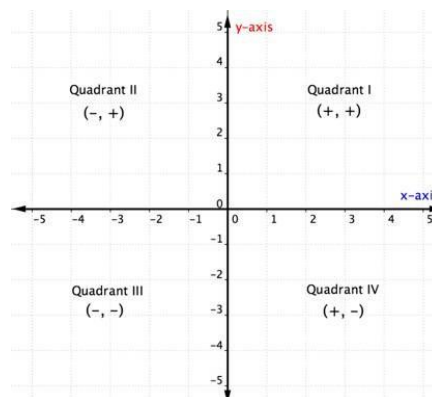
The orientation sensor uses **pitch** and **roll** for some of its measurements. The terms pitch and roll come from flight dynamics, where pitch, roll and yaw are used to determine the orientation of an aircraft. You can think of **roll as the rotation around the x axis**, and **pitch as the rotation of the y axis**.



Make an application that displays the raw data for the roll and pitch.



Add some conditional statements to this application. Using the values of roll for the x-plane, and pitch for the y-plane, add a label which displays which Cartesian quadrant the phone currently maps to:



### For students using the emulator

Note that this feature is not available for students using the emulator. This sensor will work for students using a phone via a wifi connection or USB connection.

## Pedometer

Pedometer activities can be tracked using an android device. Remember those readings we took with the Accelerometer? The pedometer uses that data, and attempts to determine if a step has been taken. It also incorporates a configurable stride length, so that it can also estimate the distance travelled.

Note that at this time, there is no MIT tutorial on this feature. However, another detailed tutorial for this feature can be found [here](#).

### *Have a go yourself!*



At the moment, the program created in the tutorial above is not so great. It allows negative input for a stride length, and gives little feedback to the user about the program status. To make the program better, and more user friendly, we want to add some conditional statements and more feedback to user.

- If the txtStringLength text field has no value, display the message "Please enter a stride length"
- If the txtStringLength text field has a negative value, display the message "Stride length cannot be negative"
- Create an identifier that informs the user if the pedometer is running
- Add control to the start and stop buttons, so that the user cannot select to start the pedometer when it is already running or stop the pedometer when it is not.

### *For students using the emulator*

Note that this feature is not available for students using the emulator. This sensor will work for students using a phone via a wifi connection or USB connection, though you may wish to take care when using the USB connection.

## Proximity Sensor

This sensor defines the proximity of an object relative to the device screen. Most proximity sensors are simply light sensors that will detect "proximity" when they are covered.

Note that this feature is not available on all android devices.

There is no MIT tutorial on this feature, however, a simple youTube tutorial can be found **here**

*For students using the emulator*

Note that this feature is not available for students using the emulator. This sensor will work for students using a phone via a wifi connection or USB connection