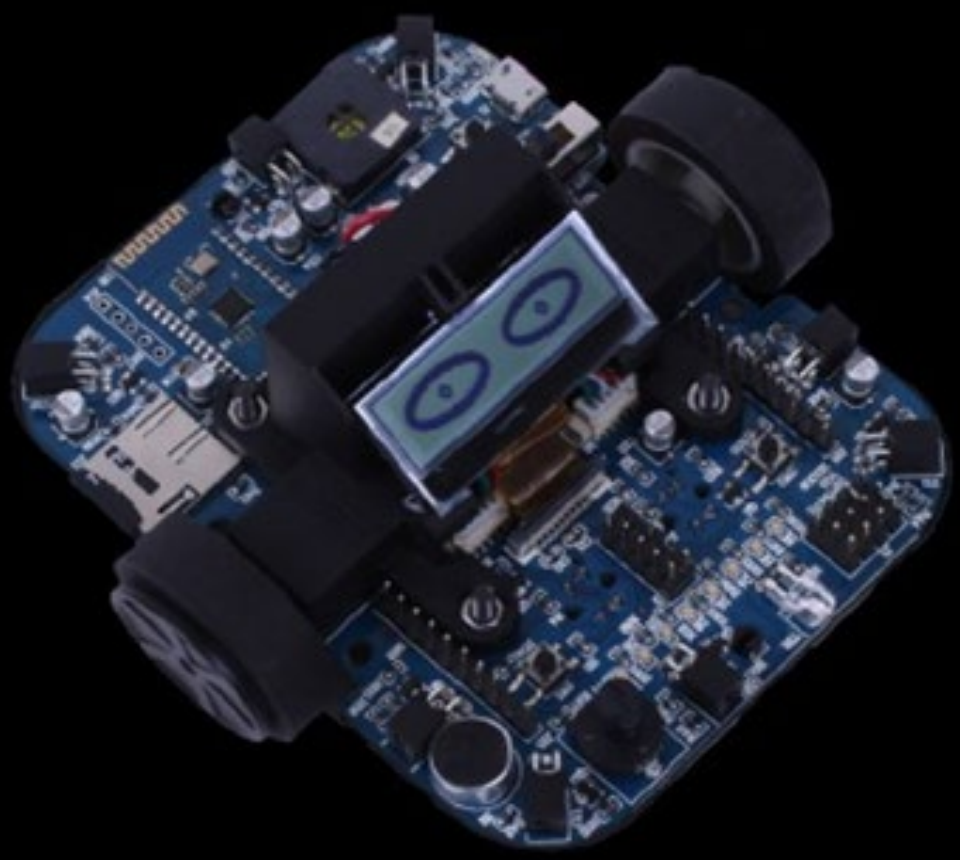
MATRIX | FORMULA ALLCODE



ROBOTICS COURSE - INSTRUCTIONAL GUIDE

CP5894

Inspiring The Next Generation Of Engineers

www.matrixtsl.com

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Contents

Introduction

Getting started 3
Take the Tour - Check out my body parts 4
Language Neutral - Application Program Interface 8
Pseudo-code - A simple programming aid 10

Hardware setup

Windows PC 11
Android Tablet / Phone 13
Raspberry Pi and Linux 15

Controlling the Robot

Using Flowcode 17
Using App Inventor 19
Using C++ / C# / VB 22
Using Python 24
Using LabView 26
Using Scratch 27

Worksheets

Worksheet 1 - Robo makes its first move 30
Worksheet 2 - Watch out - those lights are animated 32
Worksheet 3 - Switch it on, switch it off 34
Worksheet 4 - Hello world 36
Worksheet 5 - How bright is that light? 40
Worksheet 6 - Searching for the light 42
Worksheet 7 - Follow my line 44
Worksheet 8 - Push buttons do the work 47
Worksheet 9 - Status panel 49
Worksheet 10 - Tilt and turn (using a mobile device) 51
Worksheet 11 - Lefty can navigate through a maze 53
Worksheet 12 - Play that tune 55
Worksheet 13 - Robo-DJ 57
Worksheet 14 - Mobile bug 59
Worksheet 15 - Robo-Pop 62
Taking it further - Challenges 16-20 64

Appendix

API Reference Chart 65
Microcontroller Pin Connections 69
Musical Note Frequencies 70
Setting the Robot's Name 71
Reloading the AllCode API Firmware 72

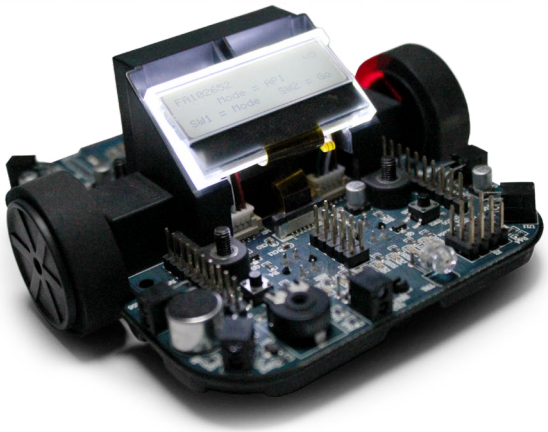
Copyright © 2016 - 2026 Matrix Technology Solutions Limited

INTRODUCTION

Getting Started

Where do you start?

Congratulations. You now have a state-of-the-art Formula. AllCode robot that can perform all sorts of interesting manoeuvres, navigate a maze, dance, make sounds, play music, record and playback audio. The list goes on and on. As there are so many things you can do with the robot, the first problem you might encounter is working out the best place to start.



This instructional guide has been prepared to help you get up the learning curve as quickly as possible so you can enjoy making the robot do the things you want it to do.

Fun

One of the key elements of this instructional guide is to make sure you have some fun. Learning or acquiring a new skill is always much more rewarding when the task has a lot of fun associated with it. The worksheets in this guide book have been developed to give you hours of fun and enjoyment while you learn the basics of robotics. Getting the robot to make its first moves, then to utter its first sounds might start off as a challenge - but should end up as pure delight and personal satisfaction when you achieve it.

Creativity

Although you can use the robot on your own, it's much more fun when you team up with one or more of your friends, as you will find that each of you have ideas and suggestions of things to try out. You will be surprised how a simple idea can be transformed by a little piece of collective creativity or imagination into something truly amazing!

Building Blocks

The worksheets in this guide are structured so they start off tackling the easy things like operating the LEDs or displaying a message on the LCD panel, then move on to more complicated tasks like navigating a maze or following a path. Each worksheet has a coloured bar that indicates the suggested "Skill Level" - easy, intermediate or advanced.

The exercises on some of the worksheets are "Standalone" which means you can dive in and try them out in any order you like. Others are "Linked" and use a building block approach that incorporates work you would have encountered in a previous activity. Again this is indicated by a coloured bar at the top of the worksheet.

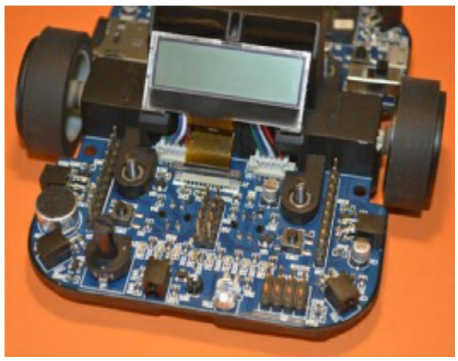
Planning your work

It's a good idea to start off by making a plan of what you want to achieve each time you use the robot. This will ensure that you are focused and productive with your time.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

TAKE THE TOUR

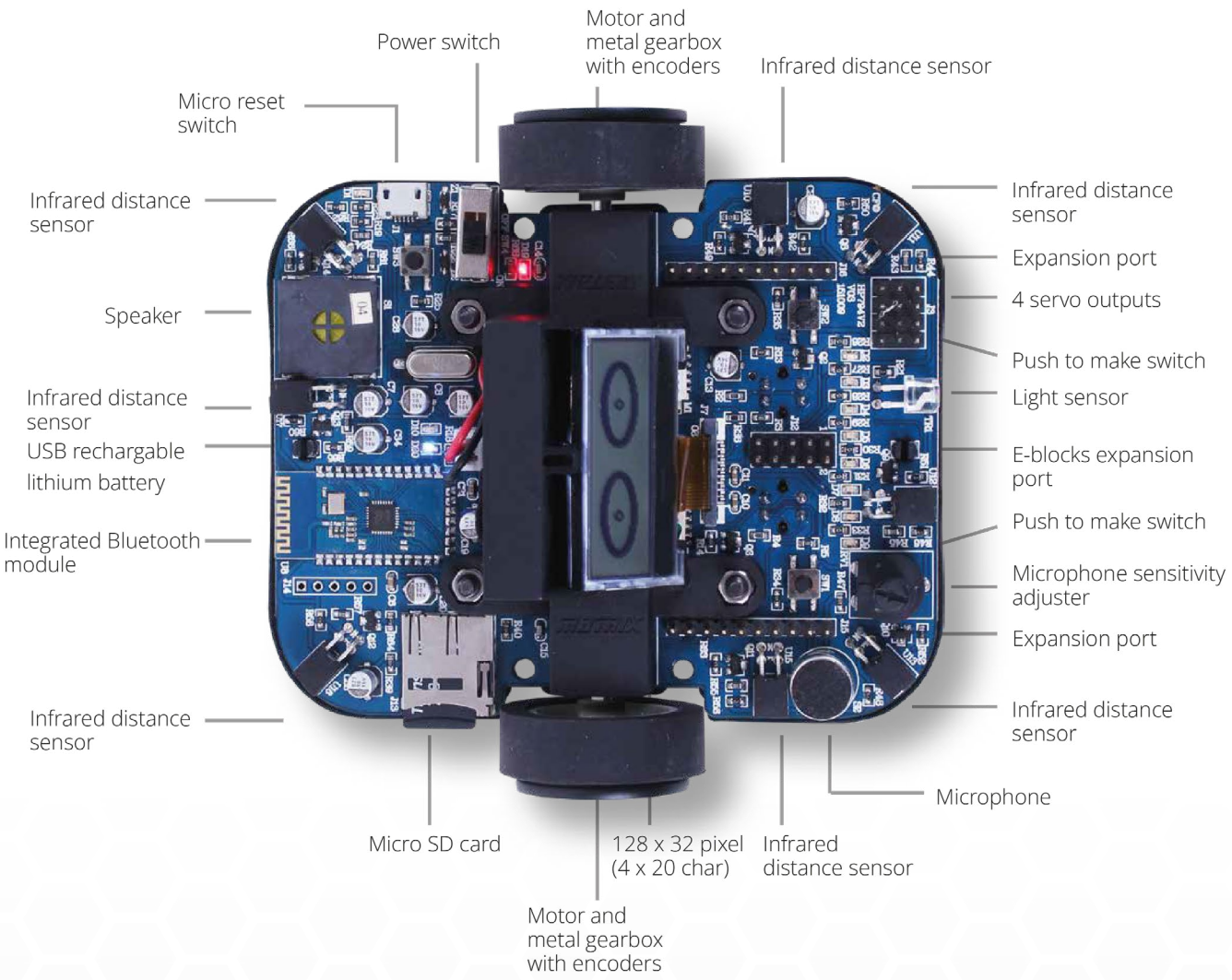
Check Out My Body Parts



This section explores the various parts of the robot and explains the basic operation of all the sensors and control systems.
Apart from the two motors that can be used to navigate the robot through a maze, there are light and distance sensors as well as sound recording equipment that will give you hours of fun.
You will probably want to revisit this page as you read this Guide.

Key parts of the robot

The block diagram below shows the important parts of the Formula AllCode robot.



Copyright © 2016 - 2026 Matrix Technology Solutions Limited

TAKE THE TOUR

Check Out My Body Parts

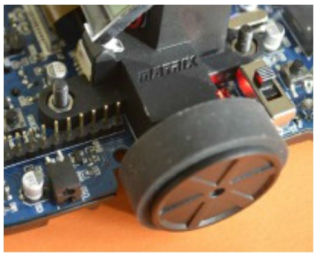
Digital Signal Controller

At the heart of the robot is a Digital Signal Controller (dsPIC®) manufactured by Microchip Technology Inc. This high performance device offers motor control (PWM), advanced analog features, a number of communication interfaces including audio capture, processing and playback as well as a considerable number of high speed counter/timers.



Motors

The robot has two powerful permanent magnet motors, coupled via reduction gearboxes (used to increase the torque) that drive rubber wheels (to provide grip on smooth surfaces). A highly efficient electronic control system ensures the power drain on the battery is as small as possible. This means you can have a lot of fun with the robot before the lithium-ion battery needs to be recharged. The two motors can be driven independently to give the robot the ability to pivot on its own axis and escape from tight or tricky situations.



Feedback encoders in the motors allows the dsPIC® to "know" exactly how far each one has turned - which allows the robot to move forward/backward a given distance or turn at a given angle.

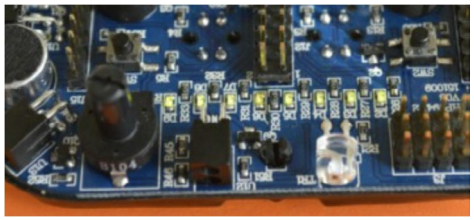
LCD Panel

Situated in the centre of the robot is a graphical LCD panel that can be used to show messages and/or simple graphical shapes.



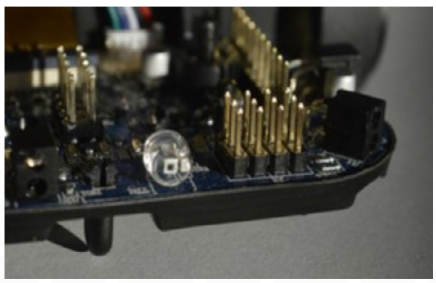
LEDs

Across the front edge of the robot is a row of green LEDs that can be driven by commands from your program.



Light Sensor

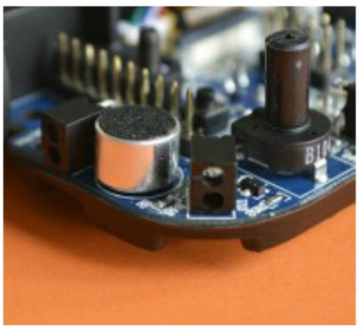
Positioned along the front edge of the robot is a light sensor. It can be used to measure and report back to your program the amount of light falling on the robot.



If this is linked to controlling the motors it enables the robot to find or avoid a light source.

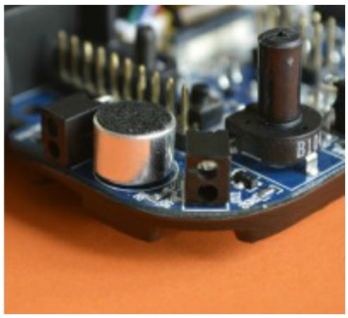
Copyright © 2016 - 2026 Matrix Technology Solutions Limited

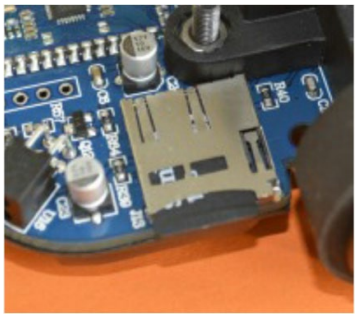
Distance Sensors



Around the four sides and four corners of the robot are mounted infrared (IR) transmitters and receivers that can measure and report back to your program the distance from the robot to a possible obstacle. Each of the eight IR transmitters can be instructed to send out a beam of infrared light. If the beam encounters an obstacle it is reflected back to the robot where it is detected by the IR receiver.

Microphone, microSD card slot and miniature speaker





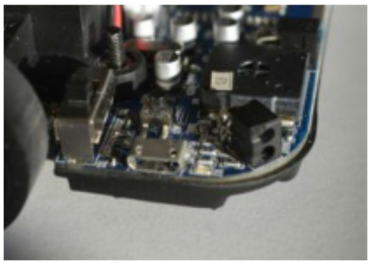
These three items work together to enable sounds and voices to be captured (using the microphone), stored on the microSD card and played back using the robot's on-board speaker.



Pre-recorded sounds and music, previously written onto the card, can also be played.

Notes generated by the robot can be played and linked together to make musical sounds.

Micro USB Connector



This connector can perform two functions. It can be used to recharge the robot's lithium battery by plugging in a charger like the one you probably use with your mobile phone. It can also be connected to a computer/laptop/tablet (with a USB socket) to download and execute programs you have written.

On-board Bluetooth Module



This is a small daughter board mounted on the rear edge of the robot. It provides another method for you to download programs to the robot.

One of its main uses is to enable you to connect wirelessly to the robot and then interact with the on-board sensors, control the motors, write messages to the LCD, operate the LEDs, and playback sounds, etc.

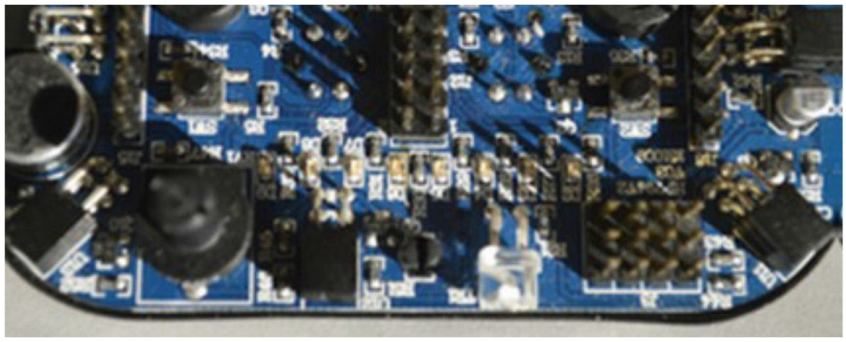
Copyright © 2016 - 2026 Matrix Technology Solutions Limited

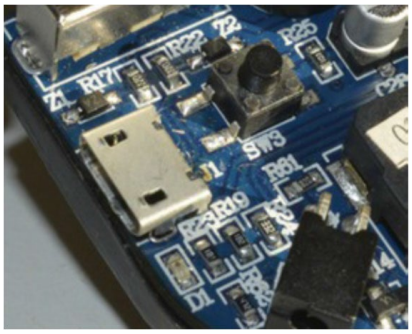
TAKE THE TOUR

Check Out My Body Parts

Push Buttons

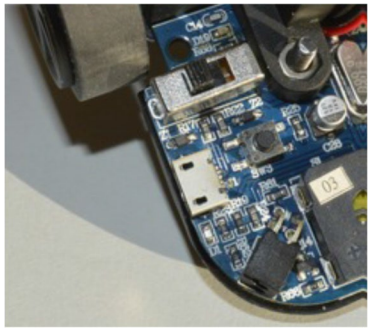
There are three push buttons on the robot labelled SW1, SW2 and SW3. The first two buttons can be sampled by your computer program to trigger a user-defined task. SW3 performs a hardware reset, to restart your program, when it is pressed.





Slider Switch

This is labelled as SW4 and is the robot's main On/Off switch.



Line Detectors

If you turn the robot over you will see two line detectors. Each unit sends out a beam of light. Depending on the colour and what sort of surface (i.e. reflective or non-reflective) is underneath the robot determines how much of the beam returns.

The line detectors are primarily used to get the robot to follow a path drawn on a piece of paper or printed on a mat.



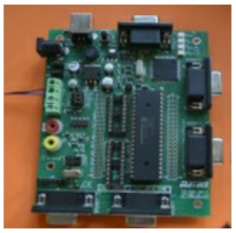
This is achieved by writing a program to detect when the robot strays off the edge of the path, adjusting the speed of the two motors to bring the robot back on track.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

LANGUAGE NEUTRAL

Application Program Interface







Language independent, agnostic, language neutral, platform independent - what do these terms mean? Basically it means you are not forced to use a certain programming language or platform to control the robot.

This is because Formula AllCode offers an Application Program Interface (API) that enables you to interact with the robot using a set of simple routines or protocols.

If you have not heard of APIs then this section will help you understand how to make use of it with the robot.

One way to explain this is to think about how a TV remote controller works. Although modern televisions have touch buttons or soft-touch areas along the edges of the screen, most people find it more convenient to use a remote controller to turn the TV On/Off, change a channel, adjust the volume or brightness settings, etc.

When you think about it, a TV remote is a very simple device consisting of a keypad and an infrared light beam. When a button is pressed its value is encoded and used to send a binary pattern, via the infrared beam, to the TV. The TV decodes the received pattern and carries out the required function.



If for some reason the TV remote failed it would be an easy task to replace it with a new one, or even purchase a universal remote (if you had a number of devices to control).

There are Apps that can be used to turn smart phones into a TV remote controller and other hardware is available that can send out TV remote codes. The only critical part is to send the correct pattern (i.e. command) when it is required.

So you could say the TV has an API that allows a remote controller (whatever form it might take) to control the intelligence or electronic control systems within the television.

The API, as used on the Formula AllCode Robot, offers the same platform and language independence as the TV example described above. The difference is the transmission medium for the robot is Bluetooth rather than an infrared beam. This means providing you have a Bluetooth facility on your system, you have the freedom to use your favourite platform and programming language to interact with the robot.

As an example, you might choose to use a Bluetooth-enabled mobile phone to control the robot. Alternatively, you could Bluetooth-enable a PC/Mac/Raspberry Pi® or a Matrix Multiprogrammer board by fitting a low-cost Bluetooth module to create a simple control system.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

As the API reacts to a simple text-based protocol (as shown below) it means you have the freedom to use formal programming languages like C, C#, C++ or Python, or graphical or icon-based languages like Flowcode, App Inventor or LabView.

The other thing to note about the API is that some commands are bi-directional. This means that a command sent to the robot could result in a value being returned. A good example of this are the infrared distance sensors fitted to the robot. An API command could be sent to a specified sensor (that effectively interrogates it) causing it to return a numerical value of the distance between the robot and an obstacle. Shown below is the general format for the robot's API.



Text string — Numeric values and/or text string

Every command starts off with a text string that identifies what the robot should do. This may be followed by one or more parameters. Depending on what you are trying to do with the robot these parameters can be numerical or textual or a mixture of both.

For example to send a value to the eight LEDs on the front of the robot you would use:

LEDWrite <value>

As the LEDs are grouped together and form an 8-bit row, they can be driven by sending a binary number to them. So the parameter <value> can take a value between 0 and 255.

It should be noted that the API commands for a particular language might have some subtle differences. For example, Python will use something like "fa.LEDWrite(20)" whereas C# would look like "FA_DLL.FA_LEDWrite(20);" and for App Inventor, Flowcode and LabView the appropriate icon would be selected.

Here's another example that shows how to control the motors on the robot.

Forward <distance>

The parameter labelled <distance> can take a value between 1 to 1000 to define the distance the robot will travel (in millimetres) in a straight line.

If you wanted to independently control the two motors you could use this command.

Setmotors <left> <right>

The parameters <left> and <right> can take a value from -100 to 100 to define the speed and direction of each motor, allowing the robot to move in various directions.

The API is listed in the appendix and worksheets in this Guide will explain in more detail.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

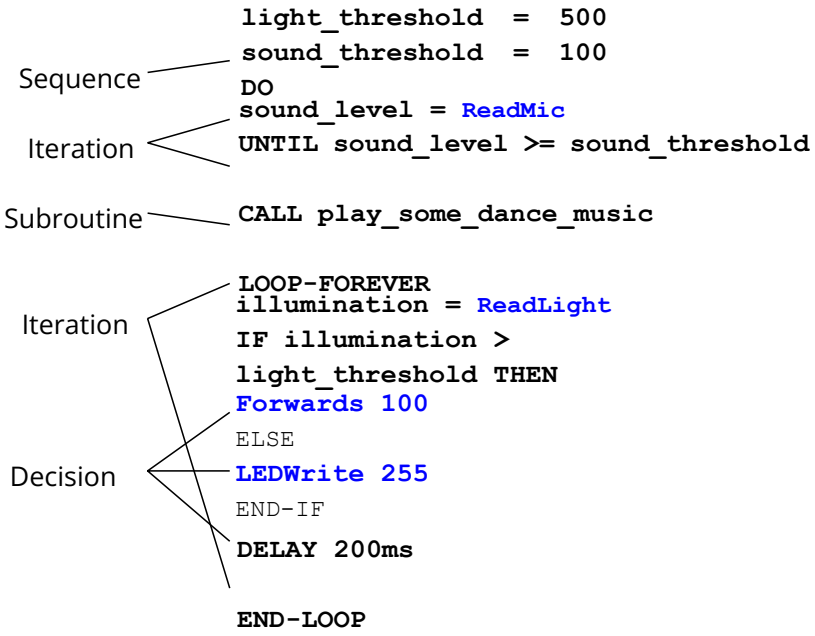
PSEUDO-CODE

A Simple Programming Aid

Most of the examples that appear in this Instructional Guide are written with pseudo-code. You may question why we didn't use flowcharts or one of the many programming languages that can be used with the Formula AllCode Robot.

The answer is twofold. Firstly, this Guide would be huge if all the examples were written in every language the robot supported, and only certain parts would be applicable to you.

Secondly, if you use flowcharts there is nothing to prevent you creating structures that do not easily map onto the three main control constructs (e.g. sequence, decision and iteration) that formal programming languages support. Here's an example of a program, written in pseudo-code, using all three constructs for the robot.



A sequence is a group of one or more statements that follow each other. There are two examples of iteration in this piece of pseudo-code: the first one repeats until a certain event occurs and the second one repeats forever. A decision, sometimes called a branch, can take one of two paths depending on the answer to a question.

Also shown in this example is a CALL that jumps to a subroutine. This is used to make the code easier to read by hiding some of the unnecessary detail.

The items highlighted in dark blue are API calls that allow you to interact with the robot. The name of these calls will map across to the set of macros available in Flowcode, App Inventor, Python and other languages supported by the Formula AllCode Robot.

Using pseudo-code allows you to take the first step of putting your ideas into practice in a structured way without getting tripped up by the syntax of a formal programming language. Once you have expressed your ideas, using pseudo-code, you can move on and write the actual program using your chosen programming language, a fairly simple coding task.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

HARDWARE SETUP

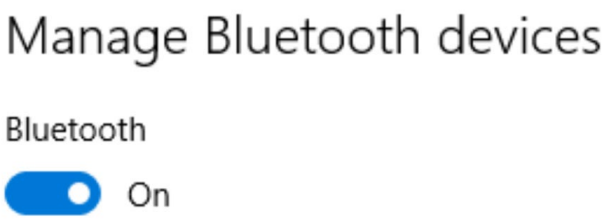
Windows PC



A lot of Windows devices, especially laptops and tablets, have inbuilt Bluetooth functionality. If your PC does not, you will need to use a Bluetooth 2 USB dongle.
Different versions of the Windows operating system use slightly different ways of connecting Bluetooth devices, but they all follow the same steps.
You will need to perform this process just once as Windows will remember which devices are paired.

1. Turn on Bluetooth

Often, Bluetooth is enabled by default and you can usually ignore this step. However if it is not, it can be enabled in the Windows settings and/or control panel. Very occasionally, Bluetooth needs to be switched on using a special switch or function-key. Please consult your PC or Windows help for more information.



2. Pair the robot

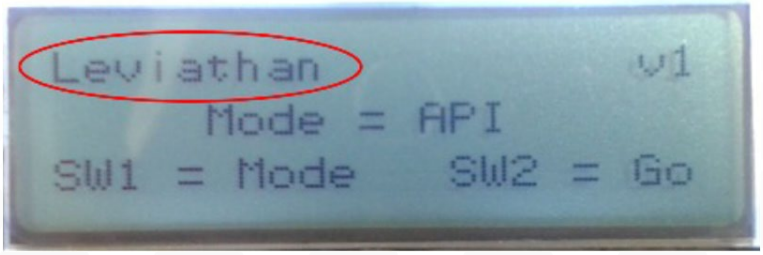
First switch on the robot - its name will be displayed in the top-left of its screen.

Again, pairing works slightly differently on the various Windows versions and so it is difficult to give specific instructions here. The Windows help and website will have guides explain how.

When pairing, you will be presented with a screen or list of available Bluetooth devices. Select the device with the name of your robot and click Next or Pair.

You will be asked to enter the pairing code. The Formula AllCode robot uses the default code of 1234, although this can be changed to another code if you want to ensure no-one else can pair with your robot.

Once the code has been entered, Windows will confirm that it has paired with the robot.

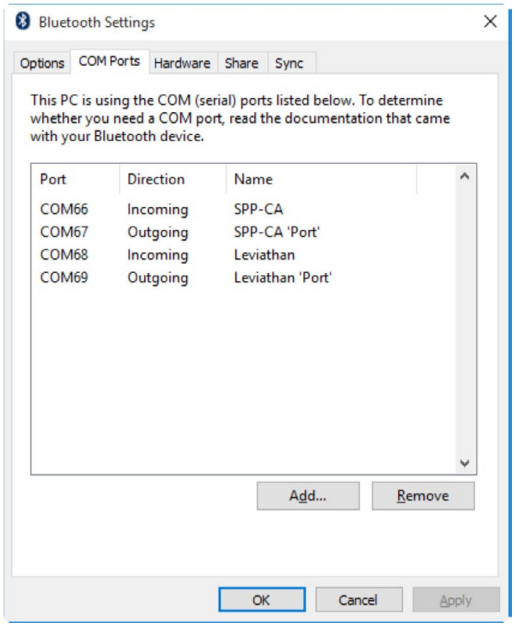


Copyright © 2016 - 2026 Matrix Technology Solutions Limited

3. Determine the COM port number

You should get a popup balloon on the task bar saying device is ready to use. If you click this before it fades away then you can find out the COM port assigned to the robot.

You use this COM port number when communicating with the Formula AllCode robot and this COM port number will stay the same as long as you do not remove or unpair the robot from Windows.

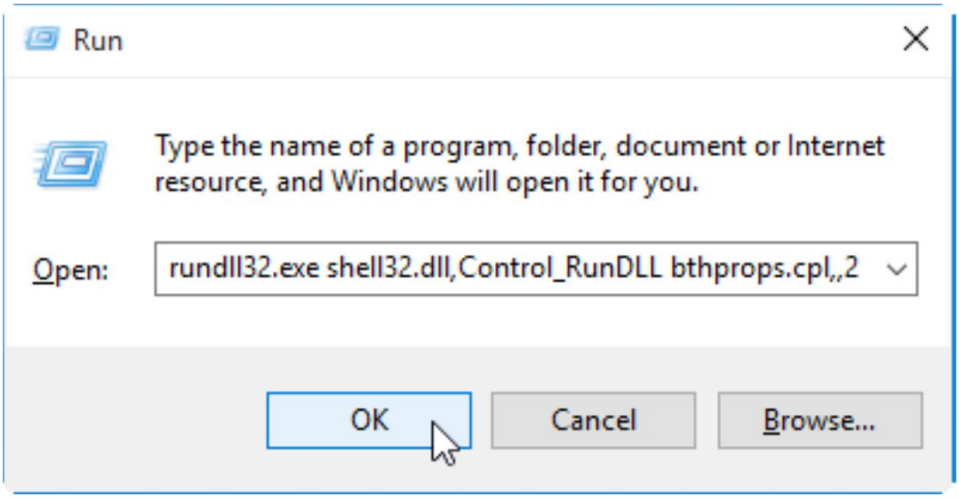


If you did not see the COM port when the robot was paired, you can find it in the Bluetooth Settings window, as shown on the right.

There are two COM ports listed for each robot. Make sure you always use the "outgoing" port number. This window can be a bit hard to find on some versions of Windows. For example, on Windows 10 you can find this via the "More Bluetooth options" link on the Bluetooth settings screen.

Luckily, there is a guaranteed way of opening this window in all versions of Windows from version 7. Open the "Run..." window by holding the Windows key and pressing R, then type (or copy and paste) the following command into the box and press "OK":

```
rundll32.exe shell32.dll,Control_RunDLL bthprops.cpl,,2
```



Now you have paired the robot and determined the COM port number, you can use any of the many programming languages available on Windows to control the Formula AllCode robot.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

HARDWARE SETUP

Android Tablet/Phone



Android phones and tablets, when used with intuitive programming software such as App Inventor, provide a motivating platform for controlling the Formula AllCode robot.

These devices almost always include Bluetooth built-in.

As with other devices, the Formula AllCode robot must be paired with the phone or tablet before it can be used.

If your Phone or Tablet already has Bluetooth functionality built in then you may first have to enable it by clicking on Settings -> Connections.

Bluetooth Switched Off



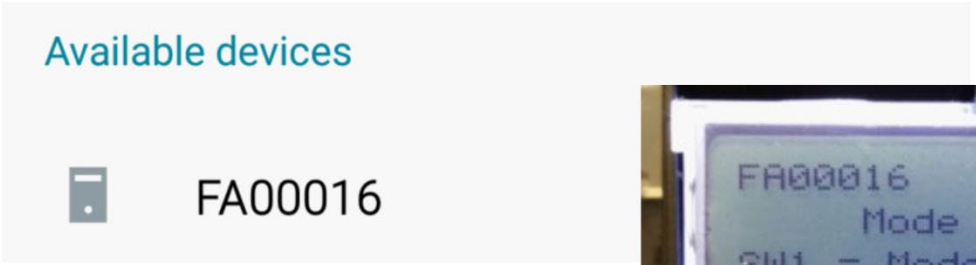
Bluetooth Switched On

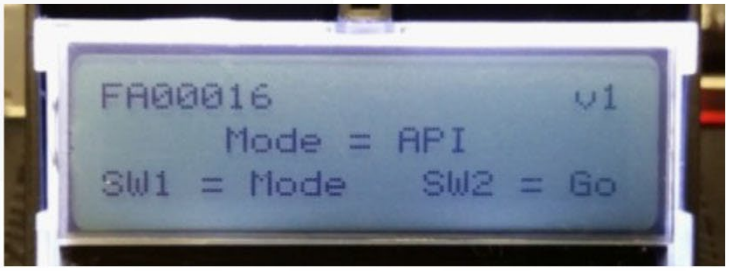


Once Bluetooth is enabled you need to pair the Formula AllCode robot to your phone to allow Apps to see the device.

Begin by clicking the Bluetooth option in Settings -> Connections.

Next make sure your robot is switched on click the Scan button on your Android device to check for new Bluetooth devices. Note you may have to scroll down to see the results from the scan.



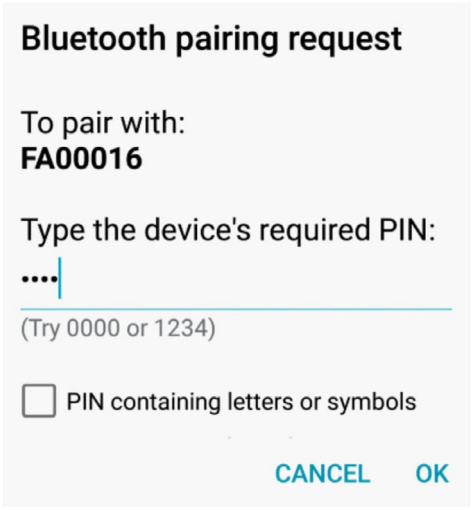


The name of the Formula AllCode appears on the top left of the LCD to let you know the Bluetooth name of the robot you want to connect to.

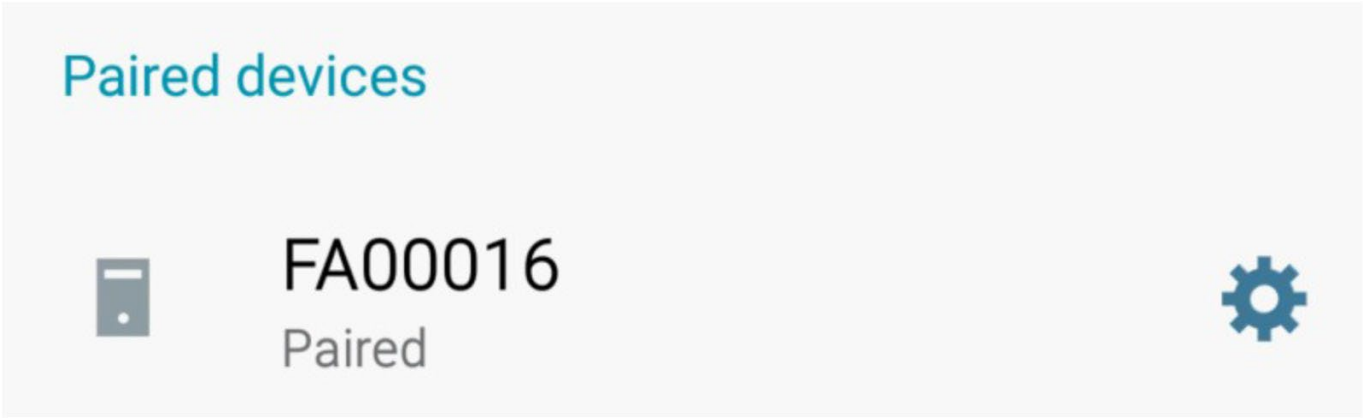
Copyright © 2016 - 2026 Matrix Technology Solutions Limited

When the device name has appeared click the device name and you will be asked to enter the pair key.

The default key is 1234.



Once the device is paired it will be listed along with any other paired Bluetooth devices you might have and is ready to be used with any Formula AllCode apps you download or create.

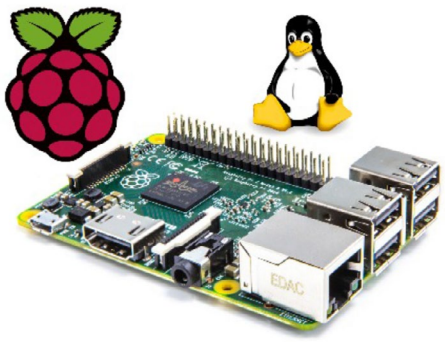


Please note: This may be subtly different on your Android device. For specifics on your Phone or Tablet please look up how to pair Bluetooth devices for your specific device.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

HARDWARE SETUP

Rasberry Pi and Linux



The Raspberry Pi is a popular single-board computer.

The most common operating system used on the Raspberry Pi is a variety of Linux called Raspbian.

The instructions here for pairing the Formula AllCode robot are not limited to a Raspberry Pi and should apply to most Linux-based computers.

Setting up Bluetooth is relatively easy on a Raspberry Pi and can be done in a number of ways. The following steps are perhaps a more complex way of setting it up, but it should work in all situations. Note the Pi needs a Bluetooth USB dongle.

Step 1 – Get your Bluetooth settings

Open a command-line terminal and type the command “hciconfig”. This will bring up a list of Bluetooth devices available on your RPi. The important thing to note is the identifier of the Bluetooth module – in my case it is “hci0”:

```
pi@raspberrypi / $ hciconfig
hci0:   Type: BR/EDR  Bus: USB
        BD Address: 00:15:83:15:A3:10  ACL MTU: 339:8  SCO MTU: 128:2
        UP RUNNING PSCAN ISCAN
        RX bytes:1420 acl:0 sco:0 events:53 errors:0
        TX bytes:452 acl:0 sco:0 commands:46 errors:0
```

Step 2 – Detect the Formula AllCode

Switch on the robot and then type “hcitool scan”. When I did this, it showed two devices. Mine was the latter (“API_B”) and you will need to take note of the 6 pairs of hexadecimal numbers that are the MAC address, a unique identifier to the robot – in my case, “00:BA:55:23:1C:20”.

```
pi@raspberrypi / $ hcitool scan
Scanning ...
        00:BA:55:23:1C:16       FormAllCode
        00:BA:55:23:1C:20       API_B
```

Step 3 – Pair the robot with the RPi

To pair, you can use the “bluez-simple-agent” command using the “hci0” and MAC address found in the previous steps.

```
pi@raspberrypi / $ sudo bluez-simple-agent hci0 00:BA:55:23:1C:20
RequestPinCode (/org/bluez/2342/hci0/dev_00_BA_55_23_1C_20)
Enter PIN Code: 1234
Release
New device (/org/bluez/2342/hci0/dev_00_BA_55_23_1C_20)
```


Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Step 4 - Making the change permanent

The final step is to make this pairing happen automatically when the RPi is next used. This can be done by editing the "/etc/bluetooth/rfcomm.conf" file (e.g. using nano) and entering the following code. Again, you will need to ensure you use the correct MAC address that was found earlier.

```
pi@raspberrypi / $ sudo nano /etc/bluetooth/rfcomm.conf
```

You will need to add a section to this rfcomm.conf file similar to the following:

```
rfcomm1 {
# Automatically bind the device at startup
bind yes;

# Bluetooth address of the device
device 00:BA:55:23:1C:20;

# RFCOMM channel for the connection
channel 1;

# Description of the connection
comment "Formula AllCode";
}
```

The three red bits of text can be customised - you will use the MAC address found in step 2, and can use and name in the "comment" field.

If you have more than one robot, you can add multiple sections - just name each one "rfcomm1", "rfcomm2", etc.

Step 5 - Testing the Connection

Once you are paired, you can test the connection by using the following in the commandline terminal:

```
echo "PlayNote 100,100\n" > /dev/rfcomm1
```

If all goes well, you should here a beep from the Formula AllCode.
If this does not work and you get "permission denied" message, you may need to add yourself to the "dialout" group. To see if this is the case, use the "id" command with your username as a parameter to check which groups you belong to. If the group "dialout" is not listed, you can add yourself to the group using the following command (remember to substitute your username in place of "username"!):

```
sudo usermod -a -G dialout username
```

You will then need to logout and log back in to see this change, and the "PlayNote" command above should now work ok.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

CONTROLLING THE ROBOT

Using Flowcode

This section explains how to use Flowcode to control the robot.

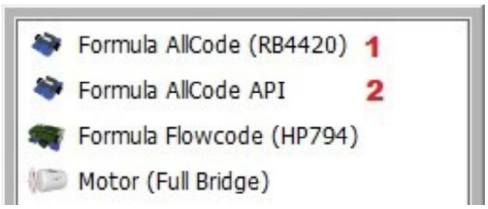
As you probably know, Flowcode provides component macros for all complex devices like CAN bus, ZigBee, and the robot. This means you can start learning about robotics and how to control the Formula AllCode Robot very quickly and very easily.

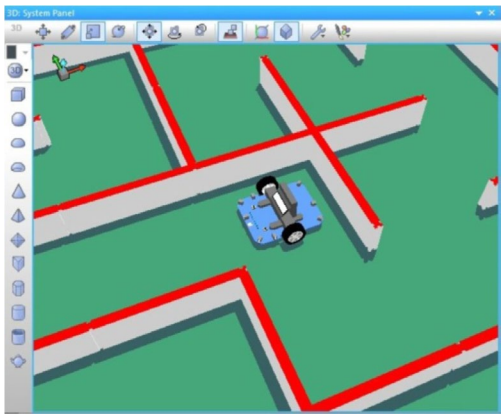
There are programs on the Matrix website to inspire and help you.

This section assumes you are familiar with the basics of using Flowcode. There are two ways of using Flowcode to control the Formula AllCode robot.

1. Re-programming the firmware on the robot
2. Using the in-built API functionality.

In Flowcode 6.1.3 and later there are two components available from the mechatronics menu to allow you to select the desired mode of operation, as shown on the right.





Downloading code to the robot

The Formula AllCode component allows us to create code which will run on the microcontroller on-board the Formula AllCode.

The component comes with a fully operational simulation allowing us to create programs to follow lines or solve mazes without having to keep compiling and downloading code to the robot.

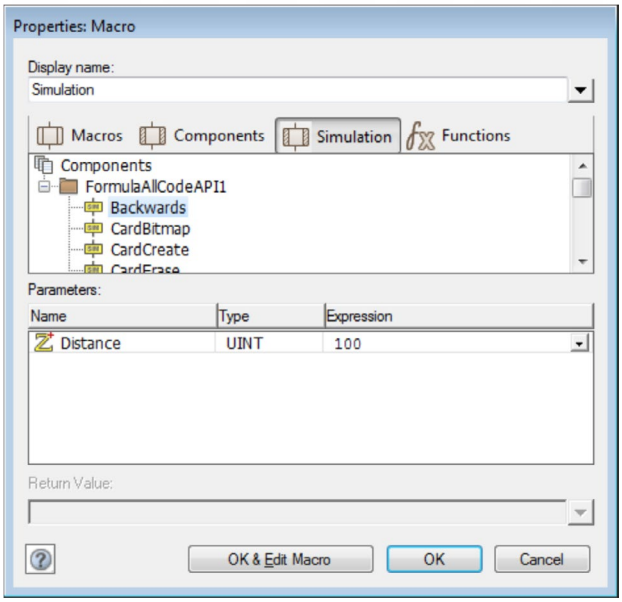
To reduce the number of macro functions within the Formula AllCode component the code for driving the Servo motor outputs, SD card and Accelerometer have not been included. There are separate components available which allow you to do this.

Note: Downloading code using this component will remove the API functionality from the robot. Instructions on restoring the API to get the robot back to the original factory firmware are provided elsewhere.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

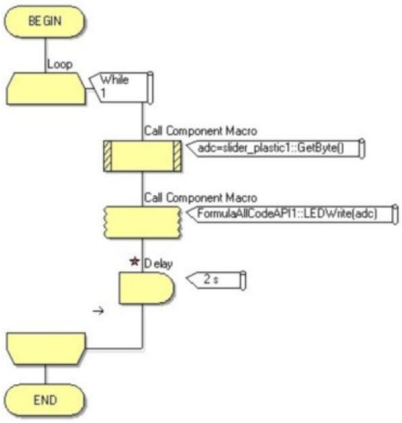
Direct control from Flowcode using the API

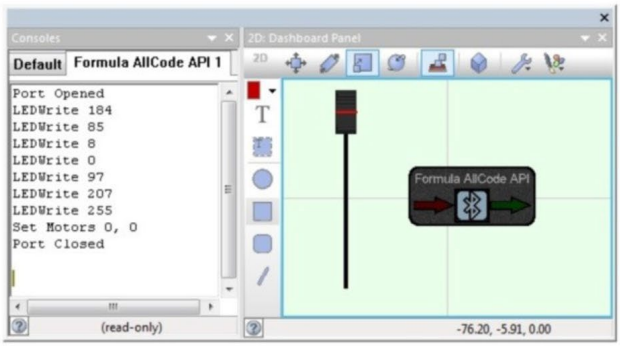
The other component ("Formula AllCode API") allows us to control a Formula AllCode robot from within Flowcode without downloading.



Note that the Component Macro functions appear in the Simulation tab rather than the usual components tab. This highlights the fact that the code is not downloadable onto Formula AllCode.

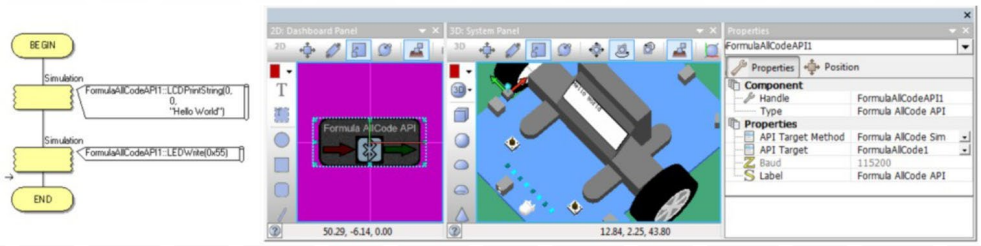
Here we control the value on the LEDs by reading a value from a simulated analogue slide potentiometer. The console now provides a list of the API function calls, the parameters and the return values.





The selected communications port is automatically opened when you start the simulation. The speed of the motors is automatically set to 0 before closing the port when you end the simulation.

By also dragging a Formula AllCode component onto the panel you can also control the simulated robot via the API component macros.



Copyright © 2016 - 2026 Matrix Technology Solutions Limited

CONTROLLING THE ROBOT

Using App Inventor

App Inventor

Example Programs

This section explains how to get started with the coding language called App Inventor that will enable you to use an Android device to control the robot. These hyperlinks will help speed-up your installation, so you can start having some fun coding.

App Inventor

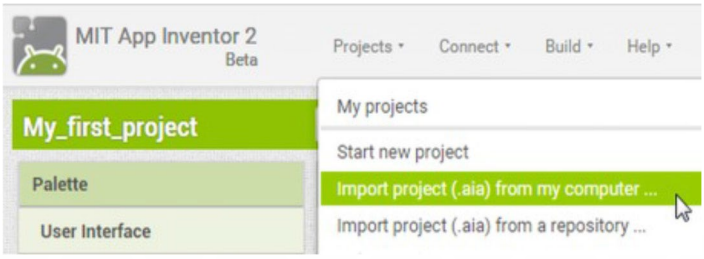
App Inventor is a freely available graphical programming language hosted on one of the cloud-computing and storage systems at Massachusetts Institute of Technology (MIT) in the United States. All you need to get started developing apps for an Android mobile phone or tablet is a web browser and a Google account. App Inventor uses colour-coded icons, shaped like jigsaw-puzzle pieces, to create an app by joining the pieces together. The system prevents you making mistakes by ensuring only certain shapes with the same colour scheme can be joined together. This technique encourages people of all ages to enjoy 'coding' and develop their confidence and ability in computer programming.

Setting up App Inventor

The key items you need are a desktop or laptop (running a modern browser like Chrome or Firefox) and a phone or tablet running the Android operating system. You will also need a QR reader so it would be a good idea at this stage to download one on to your mobile.

Just follow these simple steps to get yourself up and running really quickly.

1. Set up a Google account (if you haven't already got one).
2. Go to the App Inventor website by scanning the QR code or clicking the hyperlink above and then login using your Google account.
3. Follow the online instructions, including installing the "MIT AI2 Companion App" onto your Android device.
4. You will need to link the web-based App Inventor with your phone or tablet. To do this, select 'AI Companion' from the 'Connect' menu in App Inventor.
5. Download the Formula AllCode template onto your computer by scanning the QR code or clicking the hyperlink. Remember where you saved them on your desktop/ laptop.

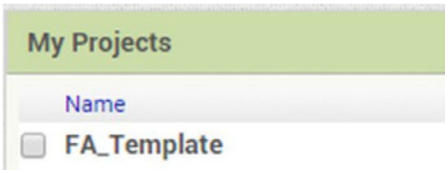


6. Import the template (you just saved) using the 'Projects' menu in App Inventor. You will use this template as a starting point for every new project.

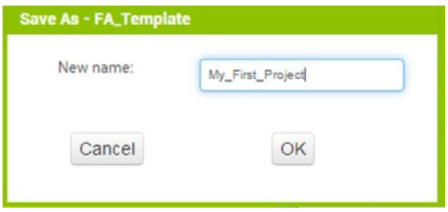
Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Your first program

Each time you want to start a new project, follow these steps:

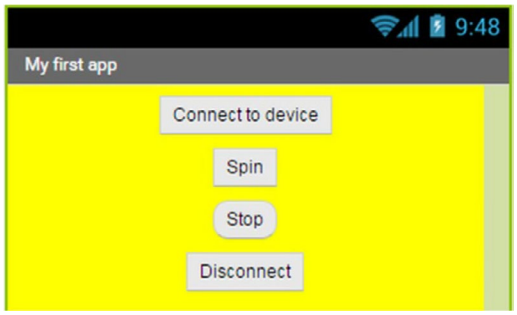


1. Load the template file by clicking 'My Projects' from the App Inventor menu and selecting 'FA_Template' project.

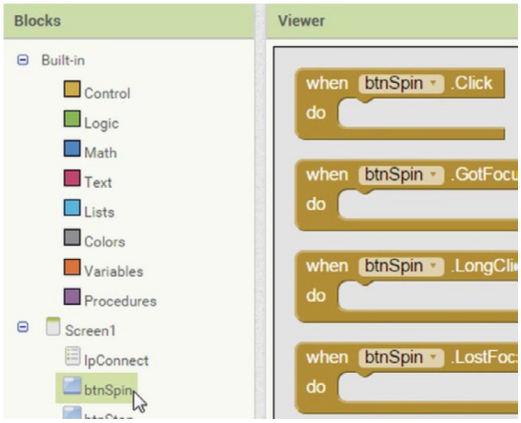


2. Save this template as a new file by selecting 'Save project as...' from the 'Projects' menu and then entering an appropriate name for your project.

3. Click 'Screen1' from the 'Components' pane and set the 'AppName' and 'Title' in the 'Properties' pane to something suitable.



4. Drag a button from the User Interface panel onto the Viewer screen and alter its text to read "Spin". Also rename the button so it reads "btnSpin". Do the same again to create another button called "Stop" with the name "btnStop".



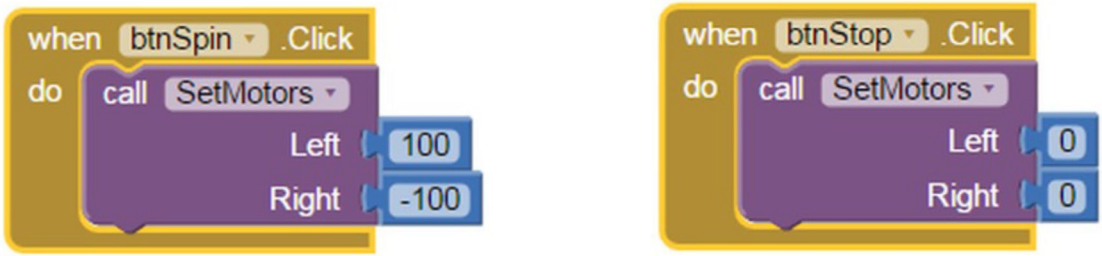
5. Switch to 'Blocks' mode and click on the 'btnSpin' object - a list of icons will appear. Drag the "when btnSpin.Click" icon onto your program.

Do the same again but this time click on the 'btnStop' object.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

6. Click 'Procedures' from the 'Built-in' list and drag the "call SetMotors" icon into the middle of your "when btnSpin.Click" and "when btnStop.Click" icons. Add two literal values from 'Math' as the Left and Right parameters to the SetMotors code blocks.

Your two blocks should look like this:

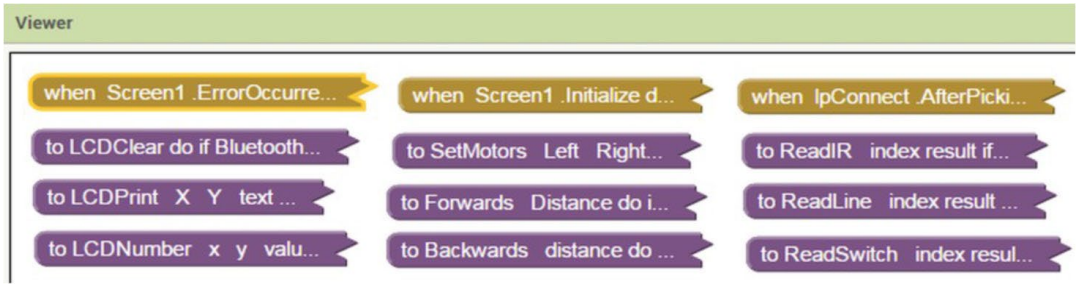


7. Now you should build the project. Select "App (provide QR code for .apk)" from the "Build' menu. Once this is complete, run the "MIT AI2 Companion" app and then scan the QR code into your Android device using the "Scan QR Code" button.

Note: If you visit the App Inventor website you will find instructions about other methods that are available for transferring your program to your Android device.

8. You can now run your program on your Android device. Click "Connect to device" and select the Formula AllCode Robot from the list. Clicking the "Spin" button should make the robot spin, and "Stop" should make it stop.

You may have noticed a number of icons at the top of the screen in App Inventor. These define the procedures for communicating with the Formula AllCode Robot and some standard functions to allow the Bluetooth link to be set-up.



The tan coloured icons represent events such as when a button is clicked, when a timer triggers or when an error occurs.

The mauve coloured icons relate to a set of procedures or subroutines that have been designed to perform certain tasks for you. You should not alter these unless you are an experienced App Inventor user.

Some of the worksheets in this Instructional Guide show how events and procedures can be joined together to carry out tasks that interact with the user and the robot.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

CONTROLLING THE ROBOT

Using C++/C#/VB



A common programming tool is Visual Studio from Microsoft.

In this section we introduce various methods to communicate with the Formula AllCode robot using some of the more widely known programming languages such as C++, C# and Visual Basic.

Using the Formula AllCode with Visual Studio via the Visual C++, Visual C# or Visual Basic programming languages is fairly straightforward and consists of using a DLL library and associated files provided by MatrixTSL to communicate with the robot. As will other languages, you need to use the COM port number that the robot is connected to. There are examples on the Formula AllCode pages of the Matrix TSL website here: https://www.matrixtsl.com/allcode/resources/

Using C#

The program on the right shows a basic program in C#.

You should use the namespace "FormulaAllCode" and place the FA_DLL library file in the same folder as your project. The DLL itself needs to be in the same folder as the EXE you create.

You will notice that the Formula AllCode API commands are prefixed with the characters "FA_" and also need to have the COM port sent to them each time as the first parameter.

Remember to modify the API commands in the appendix when using them.

Also remember to close the COM port at the end of your program!

```
using System;

namespace FormulaAllCode
{

class MyProgram
{

static void Main(string[] args)
{

//Assign the port number
char iPort = (char) 9;

//Open the COM port
FA_DLL.FA_ComOpen(iPort)
;

//Play a tune
FA_DLL.FA_PlayNote(iPort, 523,
400);
FA_DLL.FA_PlayNote(iPort, 659,
400);
FA_DLL.FA_PlayNote(iPort, 784,
800);

//Close the COM port
FA_ComClose(iPort);
}
}
}
```


Copyright © 2016 - 2026 Matrix Technology Solutions Limited

CONTROLLING THE ROBOT

Using C++/C#/VB

Using VB

The same program is shown on the right, this time in Visual Basic.

You will see that the program is is very similar to the C# program, with only some minor differences in syntax. The calls to the Formula AllCode API are identical.

The FA_API.vb file should be added to your project.

Remember also to put the "FASlave.DLL" file into the same folder as your created EXE.

```
Module Module1

  Sub Main()
    'Assign the port number
     Dim iPort As Byte
     iPort = 9

    'Open the COM port
     FA_ComOpen(iPort)

    'Play a tune
     FA_PlayNote(iPort, 523, 400)
     FA_PlayNote(iPort, 659, 400)
     FA_PlayNote(iPort, 784, 800)

    'Close the COM port
     FA_ComClose(iPort)
  End Sub

End Module
```

Using C++

A slightly different program is shown for C++, this time drawing an equilateral triangle.

To use the DLL with C++, you need to reference the functions by including the "FA_API.h" header file. You also need to add the "FASlave.lib" file to your Visual Studio project.

Also put the DLL into the same folder as the EXE you create.

As with the other languages, the calls to the Formula AllCode API are very similar, meaning it is very easy to use the robot with different languages - assuming you know the basics of that language anyway!

```
#include "stdafx.h"
#include "FA_API.h"

int main()
{

   //Assign and open the COM port
   char iPort = 9;
   FA_ComOpen(iPort);

   //Draw a triangle
   for (int i=0; i<3; i++)
   {

        FA_Forwards(iPort, 500);
        FA_Left(iPort, 120);
   }

   //Close the COM port
   FA_ComClose(iPort);

   return 0;
}
```


Copyright © 2016 - 2026 Matrix Technology Solutions Limited

CONTROLLING THE ROBOT

Using Python



Python is a widely-used computer programming language that is available on many systems. It is free, easy to learn and fun to use.

This section will show you how to set up Python for use with Formula AllCode. It is assumed you have a basic working knowledge of Python itself. If not, there are many good resources on the internet if you wish to learn this language.

Set-up

The first thing you need is to make sure Python is installed on your computer. It is usually installed by default on a Raspberry Pi, but for Windows and other devices you will probably need to download and install it from http://www.python.org.

There are two versions of Python, 2 and 3, and either can be used to control the Formula AllCode robot, but you may wish to ensure you have the latest version installed.

In addition to Python itself, you will also need to install the PySerial library. This can be found on GitHub: https://github.com/pyserial/pyserial or can be downloaded on a Linuxbased device using the following command in a terminal window:

```
sudo apt-get install python-serial
```

Now that Python and the PySerial library are installed, you should download the Formula AllCode Python library from here: https://www.matrixtsl.com/allcode/resources/

You will find examples and other resources on this page that will help you control the robot in Python and many other languages.

My first Python program

This is best shown as an example:

```
import FA
fa = FA.Create()
fa.ComOpen(5)
fa.Forwards(100)
fa.ComClose()
```

```
# Import the Formula AllCode library
# Create an instance of the API
# Open the COM port
# Move forward 100mm
# Close the COM port
```


Copyright © 2016 - 2026 Matrix Technology Solutions Limited

CONTROLLING THE ROBOT

Using Python

This very simple program drives the robot forward 10cm using the API command Forwards, but there are a number of other lines of code before and after that command that may need more explanation.

The first three lines of code import the library so you can use the API commands, then an instance of the API is created and a communication channel to it is opened. The number "5" represents the COM port that was created when the robot was paired.

It is important to close the COM port and at the end of the program we should do that using a call to the ComClose API command.

Controlling multiple robots

By creating multiple instances of the API, we can actually control more than one at the same time. The program on the right shows how this can be done.

Just like the first program, we start by importing the FA library (note we are also importing the "time" library too). We then create 2 instances of the API and open their COM ports.

The routine for drawing the square should be self-explanatory. Finally, both COM ports are closed.

Theoretically many robots can be controlled simultaneously, but unfortunately there is a practical limit due to the capability of the computer's Bluetooth device. I have found 3 or 4 is the realistic maximum.

```
# Import the libraries
import FA
import time

#Create and open 2 robots
fa1 = FA.Create()
fa2 = FA.Create()
fa1.ComOpen(5)
fa2.ComOpen(6)

#Draw a square
loop = 0
while loop > 0:
    fa1.Forwards(100)
    fa2.Forwards(100)
    time.sleep(1)
    fa1.Right(90)
    fa2.Right(90)
    time.sleep(1)
    loop = loop - 1

# Close the COM ports
fa1.ComClose()
fa2.ComClose()
```

Going further

We have shown only a few brief examples of how to control the Formula AllCode robot using Python. If you look at the API reference at the end of this document you will find many other commands that can be used.

MATRIX Copyright © 2016 - 2026 Matrix Technology Solutions Limited

CONTROLLING THE ROBOT

Using Labview



LabVIEW is a development environment for creating custom applications that interact with real-world data or signals in fields such as science and engineering.

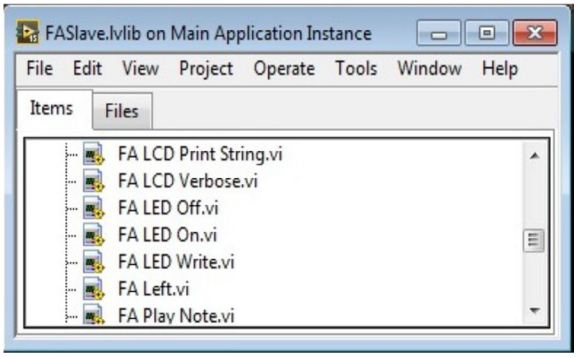
It can also be used to control the Formula AllCode robot.

This section explains how to get started.

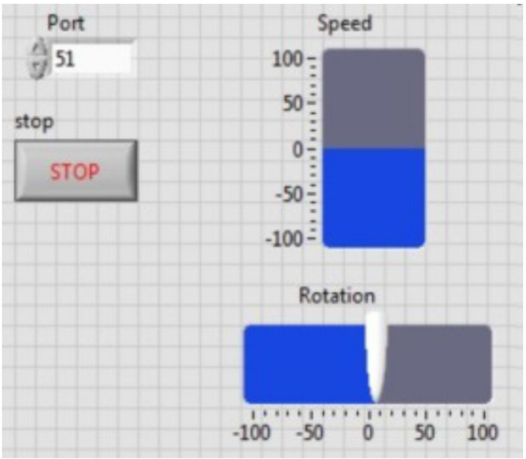
Using the robot with Labview is fairly easy and consists of using a library provided by MatrixTSL. First, download the library (which consists of a DLL and a LabView library file) from the Matrix TSL website:

https://www.matrixtsl.com/allcode/#tab4

To begin, create a new blank VI and then open the "FASlave.lvlib" file that was downloaded earlier. This contains all of the function calls to the Formula AllCode API, as shown on the right.



A sample program is shown below, using two sliders to control the robot. Set the number in the "Port" box to the COM port number created for your robot when it was paired.

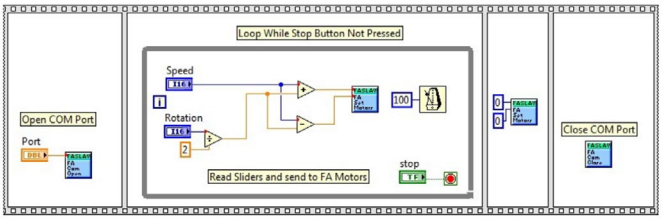


The program at the bottom of the page shows a flat sequence structure that ensures the various parts of the program are called in turn.

The left window executes first and opens the COM port.

The middle window loops until "stop" is pressed. It takes the value of the sliders and sends it to the SetMotors API command every 100ms.

The next window stops the motors when the loop has completed and the final right-most window closes the COM port.



Copyright © 2016 - 2026 Matrix Technology Solutions Limited

CONTROLLING THE ROBOT

Using Scratch

Scratch is a very popular programming tool for introducing coding techniques to school-aged children.

While predominantly used to create programs that control onscreen characters, you can also use it to control external hardware like the Formula AllCode robot.

This section describes what you need to get started.

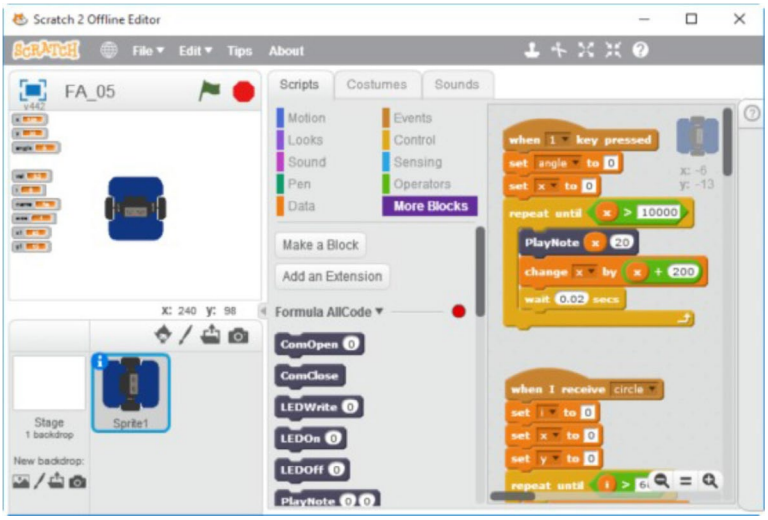
At the time of writing, there are many versions and derivatives of Scratch - both online and offline. The instructions here relate to the Scratch 2 Offline Editor.

Scratch does not natively support external hardware, so we have written an external "helper" application that translates communication between Scratch and the Formula AllCode robot. There is also a template file to get you started with using the robot. You can download this helper application and the template file from here:

https://www.matrixtsl.com/allcode/resources/

Before starting Scratch, run the helper application and select the robot you wish to connect to from the list.

You can now open Scratch. First open the template file to begin your new project. This will load all of the API commands in the "More Blocks" area of Scratch. It will also give you a Formula AllCode sprite and a basic example program.



Creating your own programs should now be simple.

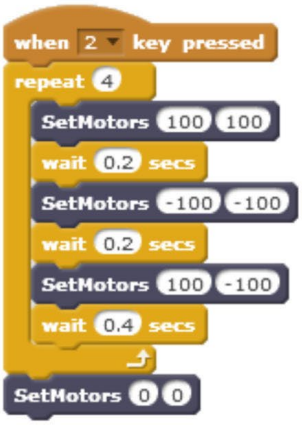
For example, to move the robot when the arrow keys are pressed, the icons below can be used.

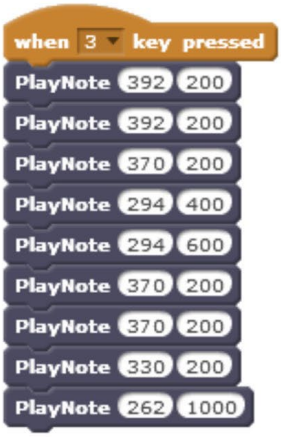


Copyright © 2016 - 2026 Matrix Technology Solutions Limited

There are many other programs you can write in Scratch. For example, here are three pieces of code:





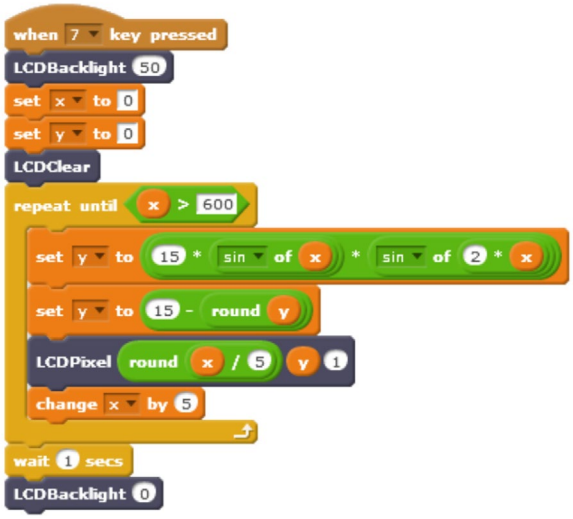


The first makes an interesting sound, the second does a little dance and the third plays a familiar tune!

Of course, you can also link with on-screen activities as shown on the right.

Here, the computer asks your name and then displays it on the LCD of the Formula AllCode robot. After that, it flashes the LCD backlight to catch your attention.





More complex programs can also be written. The program on the left draws an interesting trigonometric equation.

You can also read sensor values back from the robot and create routines that react to them.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

WORKSHEETS

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

WORKSHEETS

Worksheet 1 - Robo Makes Its First Move



This worksheet explains how to make the robot move forwards and backwards, and turn through a specified angular position so it can trace out a geometrical shape, like a square, a triangle or a house.

It's probably the very first thing you'll want to do before investigating other exciting things the robot can do.

Activity	Standalone	Skill Level	Easy

Motor Controls

There are two ways to control the robot's motors. The first method uses the API command SetMotors to set the speed and direction of rotation for each of the two motors. This particular command will be covered in detail in a later worksheet. The second method of control is based on Logo movements.

Logo movements

Logo is a language designed many years ago to teach computer programming to students, so they could control devices like a small robot or a turtle using a set of simple commands.

Forwards and Backwards

There are two API commands to control straight line movement.

```
Forwards <distance> and Backwards <distance>
```

The value for <distance> specifies the distance in millimetres the robot should travel and can vary from 0 to 1000. Here's a very simple program to make it move forwards and backwards by 100 mm.

The important thing to note about these two commands is the robot will travel in a straight line because there is rotational feedback from each motor, to control and vary the power applied to the motors. Now let's get the robot to move sideways.

```
// An example program written in pseudo-code
LOOP-FOREVER

        Forwards 100 //move forward 100 mm
        DELAY 250ms
        Backwards 100 //move backwards 100 mm
        DELAY 250ms
END-LOOP
```

Left and Right

These are the two API commands to control angular movement.

```
Left <angle> and Right <angle>
```

The value for <angle> specifies the angle in degrees the robot should rotate and can vary from 0 to 360.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

WORKSHEETS

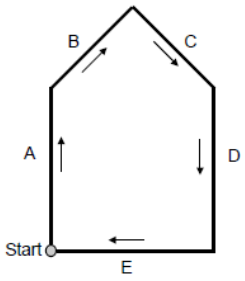
Worksheet 1 - Robo Makes Its First Move

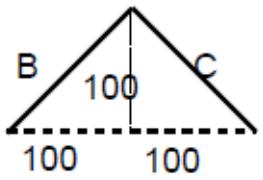
Drawing the shape of a house

Let's get a bit more adventurous and get the robot to mark out the front face of a house.

The suggested length for sides A, D and E is 200 mm.

As shown below, the roof structure consists of two rightangled triangles that have a length of 100 mm for the base and perpendicular.





Over to you:

(a) Using Pythagoras theorem you can work out the length for sides B and C. You should also be able to deduce the angles.

Then, using the program shown below as a guide, enter your calculated values (where you see the question marks) and see if the robot draws the correct shape.

```
// An example program written in pseudo-code
Forwards 200 //Draw side A
Right ??
Forwards ??? //Draw apex B
Right 90
Forwards ??? //Draw apex C
Right ??
Forwards 200 //Draw side D
Right 90
Forwards 200 //Draw base of house, side E
```

(b) See if you can get the robot to draw these geometrical shapes: equilateral triangle, right-angled triangle, hexagon, parallelogram. What other shapes can you draw?

(c) Challenge one of your friends to draw a shape and get your robot to copy it.

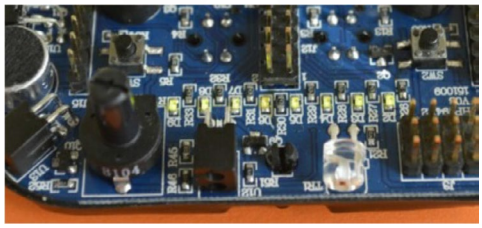
Summary

This section has explained how to use the API commands to control the robot's motors so they accurately move forwards or backwards, or turn left or right a specified angle.

The work covered in this worksheet will be really useful when you tackle other aspects of controlling the robot. For example, getting the robot to move towards the direction of a light source, reacting to a hand being placed in front of it, or navigating a maze.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Worksheet 2 - Watch Out - Those Lights Are Animated



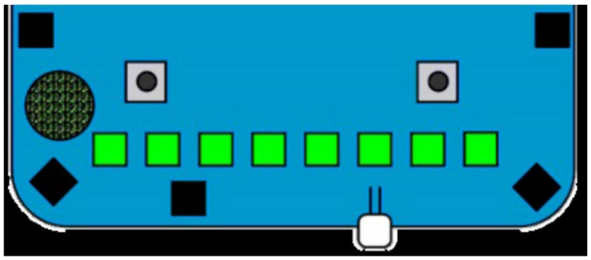
This worksheet explains how to use the eight green LEDs, located towards the front edge of the Formula AllCode Robot.

They can be used to produce interesting and creative visual effects that will enhance your enjoyment of using the robot.

Activity	Standalone	Skill Level	Easy

LEDs

At the front of the robot is a row of eight LEDs. Each one has a reference number (from LED-7 down to LED-0) as shown below.



The following two API commands enable you to turn an individual LED on or off.

LEDOn <index> and LEDOff <index>

<Index> can take a value between 0 and 7 to define the LED you want to manipulate. For example to turn LED-7 on you would use: **`LEDOn 7`** and to turn it off: **`LEDOff 7`**

Note:
These two commands just affect a single LED.

The state of the other seven LEDs remain unchanged.

This program makes LED-7 blink 4 times a second.

```
// pseudo-code example

LOOP-FOREVER
   LEDOn 7
   DELAY 125ms
   LEDOff 7
   DELAY 125ms
END-LOOP
```

If you wanted to set the state of all the LEDs at the same time then you can use the API command **`LEDWrite <value>`** where <value> can take a value between 0 and 255.

Collectively the LEDs can be represented as an 8-bit binary number (i.e. a value between 0 and 255). LED-0 is the least significant bit and has a value of 1. While LED-7 is the most significant bit and has a value of 128.

So to set the state of all LEDs at the same time you just need to work out the overall value and then use **`LEDWrite <Overall value>`**.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

For example, to light just LED-2, LED-1 and LED-0, you would need to send the value 7 (i.e. 4 + 2 + 1 = 7) as a parameter. To do this you would use the following API command.

```
LEDWrite 7
```

Over to you:

You may have heard of Knight Rider, an American television series, that featured a car with a set of headlights that were used to create an animated sequence. The best known sequence had the lights move in towards the centre and then back out again.

You should be able to create the Knight Rider pattern using the robot's eight LEDs. The first step is to work out the numerical values that need to be written to the LEDs.

The first pattern has LED-7 and LED-0 turned on together. You need to work out what value this represents. The next pattern has LED-6 and LED-1 on. Again you need to work out its value. Keeping on doing this until you have worked out all the values, then enter those values into your program as API commands.

You will probably find you need to add a delay between API commands to give a smooth transition and achieve the desired Knight Rider effect.

Other effects for you to try out
There are lots of other effects you can try out. Here are two suggestions:

The Snake pattern - this starts off with the first LED illuminated (value 1) then moves across one place to the left so that the second LED is illuminated (value 2). This sequence continues until the most significant LED is illuminated (value 128). Then the sequence retraces its steps back to the beginning. The resultant effect should look like a snake moving or weaving its head or body from side to side.

Another effect you might like to try is the Egg pattern. This pattern starts off like the Snake effect but upon reaching the top most digit (value 128) the LED remains illuminated whilst the pattern retraces its steps back to start. The effect is that of an 'egg' being laid at the far end. The sequence then repeats again until LED-6 is reached. This LED remains on (as well as LED-7) as the pattern retraces its steps.

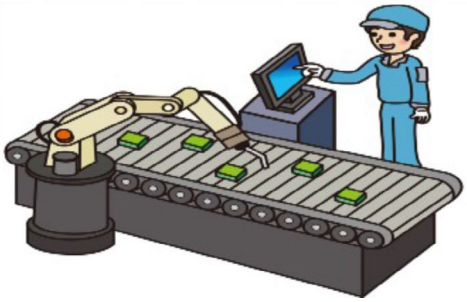
Some patterns you might want to create may follow a mathematical sequence which means you could output a value from a named variable as shown below.

```
LEDWrite <named_variable>
```

Summary
This section has shown you how to use the API commamds to manipulate the row of LEDs on the robot by sending a value to control all eight LEDs together or individually. The work covered in this worksheet will be really useful when you tackle other aspects of controlling the robot. For example, getting the robot to show the value of the light sensor.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Worksheet 3 - Switch it On, Switch it Off



This worksheet explains how the on-board push buttons and LEDs operate so that you can understand how inputs and outputs function in a simple digital system.

Activity	Standalone	Skill Level	Easy

Push Buttons

Towards the front of the robot, next to the fixings for the motors, are two push buttons labelled SW1 and SW2 that can be sensed by your program and used to trigger a task.

There is also another push button labelled SW3, adjacent to the micro-USB connector, that performs a reset function whenever it is pressed.

The API command to read the state of SW2 or SW1 is:

```
ReadSwitch <index>
```

The ReadSwitch API call uses a single “index” parameter (0 for the left button, 1 for the right button), and returns a value of 1 (pressed) or 0 (not pressed).

The way this call would be used is to assign the result to a named variable (e.g. SW_1).

```
SW_1 = ReadSwitch 0
```

If you just wanted to perform an action or a simple task when a button is pressed, then all you need to do is test if the value in the named variable is greater than zero - then carry out that particular action if the answer is true. Here’s an example program.

```
// An example written in pseudo-code
  LOOP-FOREVER
    SW_1 = ReadSwitch 0
    IF SW_1 > 0 THEN // Detect if SW1 is pressed
    CALL SW_1_TASK // User routine
    END-IF
    SW_2 = ReadSwitch 1
    IF SW_2 > 0 THEN // Detect if SW2 is pressed
    CALL SW_2_TASK // User routine
    END-IF
END-LOOP
```


Copyright © 2016 - 2026 Matrix Technology Solutions Limited

WORKSHEETS

Worksheet 3 - Switch it On, Switch it Off

Over to you:

Now that you know how to check buttons SW1 and SW2 you should try writing a program to perform a series of tasks one after another.

(a) When SW1 is pressed output the Knight Rider pattern to the robot's LEDs, and when SW_2 is pressed output the Snake pattern.

This means you will need to write some code to go inside the user routines labelled SW_1_Task and SW_2_Task to output the desired patterns.

To keep things simple at this stage, you will find that your program will perform the above sequence once, unless the appropriate button is still pressed.

When you finally release the button, the current pattern will stop when it comes to the end of its sequence.

In a later worksheet you will find out how to write a program that remembers which button was pressed, so you don't need to keep your finger on the button.

(b) Make the left button cause the robot to draw a square and the right button draw a circle.
(c) Have a look at the API commands in the appendix and see if you can make other things happen when you press a button.

Summary
This section has shown you how to read the state of buttons SW1 and SW2 and perform a certain task.

Another possible use for SW1 or SW2 is to act as a trigger, so that when a button is pressed it sends a signal to another program telling it to commence operation.

For example, you could arrange for audio (previously recorded on the micro SD card) to be played and the robot perform a dance when a button is pressed.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited



This worksheet explains how to send a text message to the LCD panel located in the central area of the robot.

The second part of this exercise covers sending a series of multi-line messages that change after a certain period of time.

Activity	Standalone	Skill Level	Easy

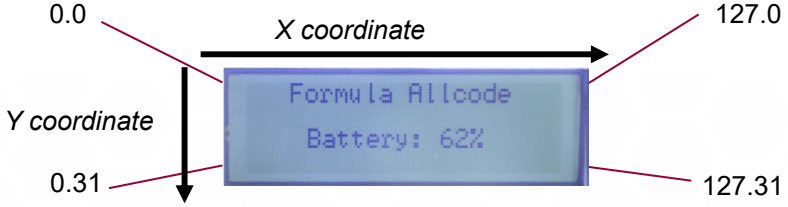
The graphical LCD (gLCD) panel, situated in the centre of the robot, is capable of drawing graphical elements like lines and rectangles, and manipulating individual pixels. Note: These aspects will be covered in greater detail in another worksheet.

Using the panel in its non-graphical mode allows messages (text and numbers) to be displayed, for example the numerical values obtained from the light or distance sensors, or just a plain message to send "visual" feedback to the user. You can of course mix the two modes by placing dividing lines, shaped blocks or user-created graphical symbols (near pieces of text) to draw the user's attention to certain parts of the panel.

The brightness of the panel's back-light can also be varied under program control. However it should be noted that having the back-light on fully will drain the battery quicker.

LCD panel layout
The diagram below shows how the X and Y coordinates of the panel are laid out. The X coordinate runs across the top of the panel (from left to right) and can take a value from 0 to 127 to define a column on the display. The Y coordinate goes from the top to the bottom of the panel and can take a value from 0 to 31 to define a row on the display.

Whenever you work in a graphical environment, it is normal practice to specify the X and then the Y coordinate (i.e. the column and then the row), so the top-left position would be known as 0,0 and bottom-right position as 127,31. A common mistake is to mix up the coordinates which means your program will display characters in the wrong place.



Characters are formed on a 6 by 8 pixel-matrix, so it is possible to display 21 characters across the width of the panel, and 4 rows deep.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

WORKSHEETS

Worksheet 4 - Hello World

Now you know how the panel is laid out, you can write a message using the API call.

```
LCDPrint <x> <y> <text>
```

Hello World
Here's an example of the message you see in most books about programming.

```
// An example written in pseudo-code
LCDPrint 0 0 "Hello World"
```

Here's an example of a multi-line message centred on the LCD panel.

```
// An example written in pseudo-code

LCDPrint 0 0 " Hello and welcome "
LCDPrint 0 8 " to the "
LCDPrint 0 16 " Robotics "
LCDPrint 0 24 " Instructional Guide "
```

Each line of the message is centred by padding either side of the text string with spaces. The number of spaces needed is calculated by subtracting the length of each message from 21 (the character width of the LCD panel) and dividing the result by two. Sometimes the result is an odd number which means one side of the message has an extra space.

Over to you:

Turn the above program into a macro or sub-routine by gathering the pseudo-code together and giving it a suitable name like MESSAGE_1. Once you have done that create other multi-line messages, give them suitable macro names and combine them together to make announcements in a timed sequence (as shown below).

```
// An example written in pseudo-code
LOOP-FOREVER
      LCDClear
      CALL MESSAGE_1
      DELAY 5s
      LCDClear
      CALL MESSAGE_2
      DELAY 5s
END-LOOP
```

Any information currently on the screen can be erased, with the API command LCDClear before sending a new message to the LCD panel.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Displaying numerical values

If you need to display the numerical value of a named variable, then this can be achieved using the API command.

```
LCDNumber <x> <y> <number>
```

Here's an example of how the value of the named variable 'distance' could be displayed.

```
LCDNumber 0 0 distance
```

To make the layout more user friendly, a piece of text could be used to label the numerical value so you know what it represents.

```
// An example written in pseudo-code
LCDClear
LCDPrint 0 0 "Distance = "
LCDNumber 66 0 distance
```

Back-light brightness

As mentioned on the previous page, the brightness of the panel's back-light can be altered to suit the viewing conditions. The API command to do this is: `LCDBacklight <value>` where <value> can vary from 0 (back-light off) to 100 (full brightness).

LCDOptions in non-graphical mode

The following API call can be used to control the way text is shown on the LCD screen.

```
LCDOptions <foreground> <background> <transparent>
```

The first two parameters can take a value of 0 or 1 to define white or black respectively. This means you can display white text on a black background, or black text on a white background. The third parameter can also take a value of 0 (display text on foreground and background) or 1 (display text on foreground only).

Related LCD commands

The following API commands affect the graphical aspect of the LCD panel and are covered in a later worksheet.

```
LCDLine, LCDRect and LCDPixel
```

Things to remember

- the layout of the LCD panel is just like a piece of graph paper
- the X axis runs across the top and the Y axis goes down the side
- the X and Y axes are numbered starting from zero (not one)

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Over to you:

This section has shown you how to use the various API commands to display and format information (text and numerical values) on the LCD panel.

(a) Get the robot to display the lines of a joke. You will need to pause before displaying the punch line!

(b) Display the results of your favourite football team. Perhaps pause between results until a button is pressed.

In later worksheets you will display useful information such as the sensor readings.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

WORKSHEETS

Worksheet 5 - How Bright is That Light?



This worksheet explains how to use the light sensor located at the front edge of the robot.

The first exercise measures the light level and displays it on the LCD panel.

The second exercise involves adjusting the illumination of the LEDs depending on the ambient light level.

Activity	Linked	Skill Level	Easy

Situated at the centre of the front edge of the robot is a light sensor. Although it looks just like a plain LED it is in fact a photo transistor that reacts to the amount of light falling on it. The amount of current that can pass through the device increases as the light increases. So, by placing the device in a simple potential divider circuit will cause a voltage to vary as the light changes. The robot can sense this and convert it into a value.

The API command to read the light level is: **ReadLight** and the way it would be used is to assign the result to a named variable (e.g. illumination).

illumination = **ReadLight**

The first thing you should do is find out the range of numerical values produced by the photo transistor under different lighting conditions. This simple exercise shows how to measure the light level and display the value on the LCD panel.

```
// An example written in pseudo-code
LCDClear
LCDPrint 0 0 "Light value = "

LOOP-FOREVER
    illumination = ReadLight
    LCDNumber 84 0 illumination
    DELAY 100ms
END-LOOP
```

Although the photo transistor can theoretically produce values from 0 to 4095, you may not be able achieve them without placing the robot in a pitch black enclosure or holding it next to a bright electric light bulb.

Write down some of the results you obtain when you place the robot in different parts of a room or take it outdoors. Make a note as to whether it is a sunny or an overcast day. You could also try shining a hand-held torch towards the light sensor and see what readings you get as you move closer to or further away from the robot.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

WORKSHEETS

Worksheet 5 - How Bright is That Light?

The next exercise is a very simple control or feedback system. The objective is to measure the ambient light level and then use this value to adjust how many of the LEDs on the front of the robot are turned on. The idea is to turn more LEDs on as the light level reduces, and turn more off as the light increases.

If this idea was applied to a practical situation (using more powerful lights) then it should be possible to maintain a room's illumination at a constant level regardless of light levels. The pseudo-code below shows how this is achieved.

```
// An example written in pseudo-code

LOOP-FOREVER
  illumination = ReadLight
  illumination = Round(illumination / 16)
  LEDs = 255 - illumination
  LEDWrite = LEDs
  DELAY 100ms
END-LOOP
```

The program begins by reading the value of the light sensor into the named variable 'illumination' and then divides this value by 16. This will scale the light reading to fall within the range 0 to 255 rather than 0 to 4091.

To make sure the result is an integer we use the Round() function. Your chosen programming language will have this function, but it might be called something else and might require an extra parameter to specify the level of rounding.

The next part of the program creates the inverse proportional lighting effect by subtracting the illumination value from 255. So what happens is, as the illumination value increases the value in the named variable 'LEDs' decreases.

The last part of the program writes the value in the named variable 'LEDs' to the LEDs on the front of the robot, then after a short delay the whole sequence is repeated.

Over to you:

This section has shown you how to use the API call to read the value of the light sensor. Try out the two exercises described here to gain experience of using the light sensor and to reinforce your knowledge of operating the LCD panel and LEDs.

The second exercise will not work as expected - if the value of the variable "LEDs" is 127, 7 LEDs will light, yet if it is 128 then only one will light. Try to change the program so the number of LEDs that light increased when the light level falls.

The work covered in this worksheet will be really useful when you tackle other aspects of the robot. For example, getting the robot to move towards the direction of a light source.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Worksheet 6 - Searching for the Light



This worksheet brings together the work covered during a couple of previous exercises to solve a real-life problem.

The problem that needs to be solved is for the robot to follow a light source from a hand-held torch. It sounds simpler than you think!

Activity	Linked	Skill Level	Intermediate

Strategy

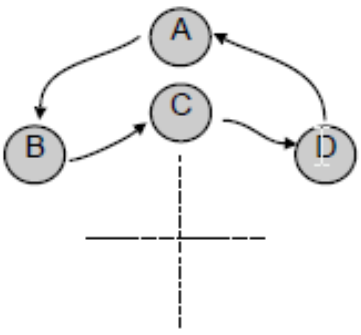
The first thing you need to do, to solve the problem outlined above, is to work out a strategy for dealing with the possible situations that could occur. For example, if the robot detects the light-beam from the torch, then move forward a certain distance and check again. Pretty obvious, but what should the robot do if it loses sight of the beam? It needs to stop and try to find the light source.

One way to work out what the robot should do is to put yourself in the same situation and think what you would do. Using your eyes you could probably see the light-beam. You might have to move your eyes slightly to the left or right in order to 'see' the beam again. If the beam has dramatically changed its position, then you might have to move your head left or right because the amount of travel your eyes can make is physically limited.

The anatomy of the robot is a little bit different from ours. It might only have a single eye (i.e. light sensor) in a fixed position (on the front edge of the robot), but it is able to move its head through a full 360 degree rotation. Something we can't do!

This is the strategy to try to find the light-beam. Currently the robot is stationary, so make it rotate left by a certain angle, say 5 degrees, and check for the beam. If it manages to find the beam then it starts moving forward. If it is unsuccessful, then it moves its head to the right by the same angle, and checks again. If these two manoeuvres were unsuccessful, then the angle is increased, to say 10 degrees, and the procedure repeated.

Point A is the robot pointing straight ahead. The arrow from A to B indicates an angular movement to the left. B to C represents the robot returning to point forward again. The arrow from C to D indicates an angular movement to the right, while D to A is the return movement to point ahead.



The existence of the light-beam is checked at all four points. The transition from D to A is the place where the detection angle is increased before the sequence is repeated.

If detection angle reaches 180 degrees this will indicate the light-beam failed to be located.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Here's an example of the structure for the light searching program. To assist you in reading through the program the API commands have been highlighted in blue.

```
// An example program written in pseudo-code

threshold = 2000 //Experimental value
pos = "A"
movement = 5 //Initial 5 degrees
LOOP-FOREVER
  illumination = ReadLight
  IF illumination > threshold THEN
     pos = "A"
     movement = 5
     Forwards 100
  ELSE
     IF pos == "A" THEN
       pos = "B"
       Left movement
     ELSE-IF pos == "B" THEN
       pos = "C"
       Right movement
     ELSE-IF pos == "C" THEN
       pos = "D"
       Right movement
     ELSE //Must be at point D
       pos = "A"
       Left movement
       IF movement < 180 THEN
        movement = movement + 5
       ELSE
        movement = 5
       END-IF
     END-IF
   END-IF
   DELAY 50ms //Short delay (see text)
END-LOOP
```

Although the structure closely matches the textual description, given on the previous page, there are a few things that you should note. The threshold value was obtained by experimenting with a small hand-held torch in a room with subdued lighting. You need to aim for the situation where the amount of available light plus the light from the torch does not exceed the upper limit of the light sensor (e.g. 4095).
The main loop includes a short delay to smooth out the timing of the left/right movements. You could try changing the value of the named variable movement and see how this affects the behaviour of the robot and its ability to locate the light-beam.

Over to you:

This section has shown how to combine the API calls for the light sensor with Logo-style movements to solve a real problem. See if you can program your robot to do the same.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

WORKSHEETS

Worksheet 7 - Follow My Line



This worksheet brings together some tasks covered in previous sections, to solve a realworld problem.

The problem that needs to be solved is for the robot to use the on-board sensors to follow a black line on a white background. It sounds simpler than you think!

Activity	Linked	Skill Level	Intermediate

If you've not looked underneath the robot now is the time to do so. You should be able to spot the two sensors that detect the amount of light reflected off of a surface. Each sensor includes an infrared emitter and a photo transistor, that can identify the difference between white and black surfaces based on the contrast and reflective properties of an object.

The API command to read the line sensor is:**ReadLine <index>**and the way it would be used is to assign the result to a named variable (e.g. line_reading).

The parameter **<index>** can take a value of 0 or 1 to select the appropriate sensor.

Here's an example of reading line sensor-0 which is the left-hand sensor when viewed from the upper side of the robot with the face of the gLCD panel pointing away from you.

```
left = ReadLine 0
```

As an exercise you could write a simple program to read the line sensors and display their values on the LCD panel. This will enable you to experiment and see what sort of values are returned from different coloured surfaces.

```
// An example written in pseudo-code
   LCDClear
   LCDPrint 0 0 " Left Right "

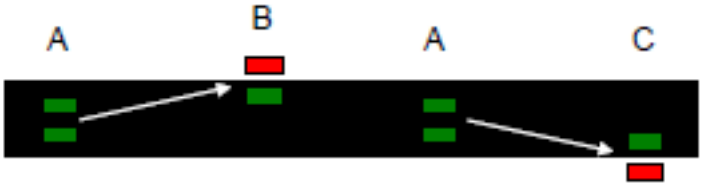
LOOP-FOREVER
     left = ReadLine 0
     right = ReadLine 1
     LCDNumber 24 0 left
     LCDNumber 72 0 right
END-LOOP
```

The above program should produce a clearly formatted screen with the line sensor values displayed under the headings Left and Right. This will enable you to record the reflection values for different types of surfaces. Note the readings for a black and a white surface. The sort of readings you should get are probably 0 for black and 200 for white.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Setting up a test track and solving the problem
If you have the mat that came with your robot you can make use of that, or you can make your own track using a roll of 25 mm wide black tape stuck onto a piece of stiff white card. An oval or figure-of-eight track are good shapes to start off with. Make sure the bends are not too sharp or tight otherwise the robot will have a hard time navigating them.

When you use Logo movements the robot should travel in a straight, however there are factors like skidding or grease marks on the surface that could cause the robot to gradually veer off the track.



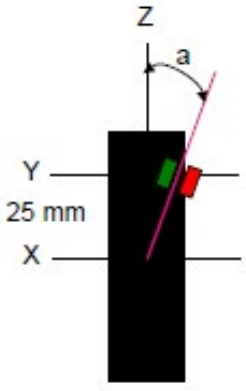
The diagram above shows what happens when the robot veers off the track. At point B and C the readings from the left and right line sensor respectively have changed to indicate the robot is straddling the edge of the track.

The diagram below show the robot moving in a straight line when the direction of track changes. Just like the above, this situation can be detected by sensing the change in the values obtained from the line sensors.



To get the robot back on track it needs to perform a right movement at point B and a left movement at C. The angle through which the robot needs to move can be obtained by trial and error, or by applying some simple geometry and trigonometry as shown below.

Point X represents the centre-line of the robot where the two wheels are fitted. Point Y is 25 mm towards the front of the robot and is the normal resting place for the line sensors when the robot is moving straight ahead. Point Z is the front to back centreline of the robot and also mid-point of the black tape. The distance between the two sensors is 17 mm which fits comfortably across the 25 mm wide black tape. Angle 'a' can be calculated by simple trigonometry or by constructing a scale drawing. Either way the result you will obtain will be approximately 10 to 12 degrees. This is the angular correction factor needed to get the robot back on track.



Copyright © 2016 - 2026 Matrix Technology Solutions Limited

If you put all these facts together you can create a program to solve the problem.

```
// An example program written in pseudo-code
black = 0            //Reflection value from a black surface
white = 200         //Reflection value from a white surface
correction = 10     //10 degrees
LOOP-FOREVER
   left_sensor = ReadLine 0
   right_sensor = ReadLine 1
   IF left_sensor >= white THEN
      Right correction
   ELSE-IF right_sensor >= white THEN
      Left correction
   ELSE
      Forwards 10 //Both sensors detecting a black surface
   END-IF
END-LOOP
```

The IF-THEN-ELSE construct checks where the robot is located. If it is on top of the black tape then it should move forward. If it's partly over the white area then the correction factor should be applied to bring the robot back to the centre.

Over to you:

Using the above program as a guide, you should try to get your robot to follow the line using Logo movements.

The program uses global variables that are set at the start of the program. This will enable you to change the values if the reflection properties of your surface are different.

You should also try changing the correction value to see how it affects tracking accuracy. Then you could explore the other way to control the robot using the API call **`SetMotors.`** This call continually operates the two motors as follows.

`SetMotors <left> <right>`

The parameters <left> and <right> can take a value from -100 to 100, where -100 means maximum rotational speed in the anticlockwise direction, and 100 means full speed clockwise. As an example, the API command **`SetMotors 50 50`** would drive the robot forwards at half speed, and **`SetMotors -50 -50`** would drive it backwards, again at half speed.

Modify the sample program, shown above, to make it work using the **`SetMotors.`** command. The speed-setting for the motors is a bit of a compromise between moving along the track at a good pace, and not going too far off-track before the line sensors detect and rectify this situation.

Using the above programs as a starting point, conduct some experiments to see how fast you can get your robot to lap the track. Challenge your friends to see who has the fastest and most accurate robot.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

WORKSHEETS

Worksheet 8 - Push Buttons do the Work



This worksheet explains how the on-board push buttons can be used to execute a specific task.

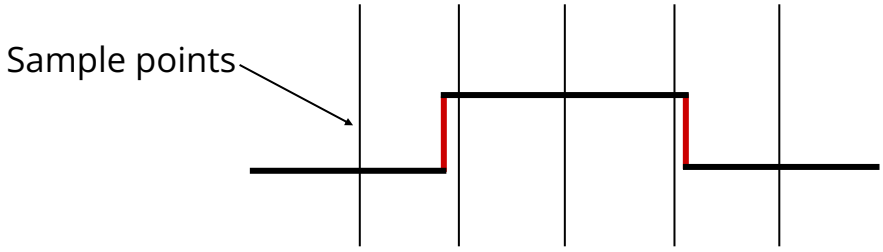
The worksheet includes an example of how to create a Up/ Down binary counter driven by the push buttons.

Activity	Linked	Skill Level	Intermediate

A previous worksheet explained how to use the API call Read <switch> to sense the state of SW1 or SW2 and execute a task if a button was pressed. Because the program was very simple it had a number of limitations. The main one being you had to keep your finger on a button if you wanted the task or sequence to repeat continuously. What if you just wanted a task to be executed once, regardless of how long you pressed the button?

It's like pressing a key on a mobile phone or a calculator. If you kept that key pressed you would only want it to be sensed once, not repeatedly, otherwise you would get a series of repeated numbers - like a machine-gun action.

What you need to do is detect the actual point when a button is pressed or released, rather than the fact that a button is pressed. These points are coloured red on the diagram below and are known as the rising and falling edges.



Although there are many ways to detect these edges, the method described here is based on taking a series of samples or snapshots with respect to time. Each sample is compared with the previous one. If they are the same then the button has not moved, and if they are different then the button has moved. The pseudo-code below shows how this is achieved.

```
// An example written in pseudo-code to detect rising-edge of SW1
   SW1_previous = 0

   LOOP-FOREVER
      SW1 = ReadSwitch 0
      IF SW1 > SW1_previous THEN        // Detect rising-edge
          CALL specific_task            // SW1 rising-edge task
      END-IF
      SW1_previous = SW1
END-LOOP
```


Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Over to you:

You now have a simple program to detect the rising-edge when SW1 is pressed. Modify the program to make the robot turn right by 90 degrees each time SW1 is pressed.

Once you have achieved that task, extend the program so that the robot turns 90 degrees to the left whenever SW2 is pressed.

If you think about the products/equipment you use that have buttons, a keypad or a keyboard, you will find that most of them do something when you press a button. What if you wanted a device to do something when you release a button?

The solution is very simple. All you have to do is modify your original program to detect the falling-edge rather than the rising-edge. This is shown below.

```
// An example written in pseudo-code to detect falling-edge of SW1
   SW1_previous = 0

   LOOP-FOREVER
     SW1 = ReadSwitch 0
     IF SW1 < SW1_previous THEN        // Detect falling-edge
       CALL specific_task              // SW1 falling-edge task
     END-IF
     SW1_previous = SW1                // Copy new sample to previous one
END-LOOP
```

You now know how to write a program to detect the point when a button is pressed and when it's released. Depending on what you are trying to achieve, you might want to detect both edges or just one. The most common situation is to detect the rising-edge.

Challenge

Write a program that uses the LEDs on the front edge of the robot to behave just like a binary Up/Down counter. The program is controlled by SW1 and SW2. Each time SW1 is pressed the binary counter is incremented, and when SW2 is pressed it is decremented.

You can decide, as it's your program, which edge will be the active edge.

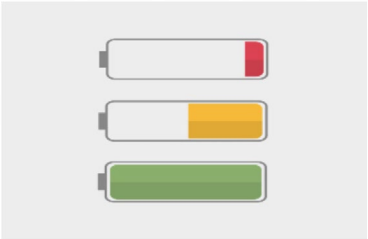
Some of the things you will need in your program are:
A named variable (e.g. counter) that will be used to hold the value of the binary counter.
An IF block to handle detecting when SW1 is pressed.
An IF block to handle detecting when SW2 is pressed.
The API command to write the value of the counter to the robot's LEDs.

After you have read some of the other worksheets in this Instructional Guide you might want to return here and combine the robot's ability to play audio, display messages or show graphical shapes on the gLCD panel with the trigger-action of the push buttons.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

WORKSHEETS

Worksheet 9 - Status Panel



This worksheet brings together a number of tasks covered in previous sections of the guide book.

The objective is to detect the status of the robot's sensors and display the values on the robot's graphical LCD (gLCD) panel or a mobile phone.

Activity	Linked	Skill Level	Intermediate

Design strategy

As the gLCD is quite compact it will be difficult to show a lot of information at the same time, so one solution is to show information for a particular type of sensor for a few seconds and then move on to the next sensor. This means you can plan each layout and then combine them together so they are shown cyclically. At the end of this worksheet are instructions on how you could make the process user-driven rather than free-running.

Planning your first layout

The first status panel to tackle is the display of the light sensor's value. Here's a sketch of how this could be laid out using a mixture of graphical lines, text and a numerical value.

Light Sensor

123

The first line is a simple static text-header consisting of 5 letters, a space followed by 6 letters making a total of 12 characters. This equals 72 pixels (i.e. 12 * 6 pixels). To get the header centred you need to subtract 72 from 128 and divide the result by 2 (i.e. 28 pixels). This means the header will start at X,Y coordinate 28,0. Here's the API call to do this.

```
LCDPrint 28 0 "Light Sensor"
```

Next you need a dividing-line that runs the full length of the panel. Rather than placing it directly under the text, it is suggested you leave a blank row of pixels so the text stands out. Here's the API command to place the line on row 9 (the tenth row).

```
LCDLine 0 9 127 9
```

The value obtained from the Light Sensor is displayed on the next line. As the value can go up to 4095 (i.e. 4 digits), this means the width to accommodate the number is 24 pixels. Again you can centre this on the panel by subtracting 24 from 128 and dividing the answer by 2 (i.e. 52 pixels). Here's the API command to display the value of a named variable.

```
LCDNumber 52 11 <named_variable>
```

Please note the Y coordinate is set to 11 as this will leave a blank row of pixels after the dividing line so the numerical value stands out. Finally you can draw another horizontal line after the Light Sensor value to complete the layout.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Over to you:

First thing for you to do is create this simple program for the Light Sensor status screen.

```
// An example written in pseudo-code
// Status panel - Light Sensor
LCDClear
LCDPrint 28 0 "Light Sensor"
light_level = ReadLight
LCDLine 0 9 127 9
LCDNumber 52 11 light_level
LCDLine 0 19 127 19
```

You should tidy things up by putting the above program in a macro. (E.g. panel_1) Then modify the code you just created for panel_1 to make panel_2 for the two Line Sensors. Carry on and make additional panels for the set of distance sensors. Then combine them together with a 5 second delay to make a rolling display just like you did with Knight Rider. The program you have just created will display each status panel for 5 seconds and then automatically move on to the next one. Although this is acceptable, it can be made much more useful and user-friendly by adding a few extra touches.

Line Sensor	
Left	Right
10	46

For example, you could use the two push buttons (SW1 and SW2) to step forwards or backwards to show the information on the various panels. This effect can easily be achieved using the rising-edge detection technique covered in an earlier worksheet.

The benefit of this approach is once a status panel is selected its information will remain on the screen until the user presses a push button. This is very useful when a lot of information, like the data from the eight distance sensors, needs to be displayed.

Another way you could navigate around the panels is to detect a sound coming from the robot's microphone. For example, if you clapped your hands this action could be detected and used to select the next panel in a rotational sequence. A worksheet towards the end of this guide book explains how this can be achieved.

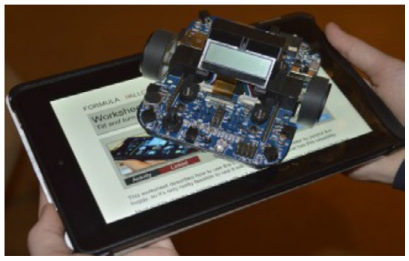
Summary

This section has brought together a number of topics covered on previous worksheets to create a really useful Status Panel. You have also discovered how to display information on the gLCD panel on a fixed-time basis, or by using the robot's push buttons.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

WORKSHEETS

Worksheet 10 - Tilt and Turn (Using a Mobile Device)



This worksheet brings together a number of tasks covered in previous sections of the guide book.

The objective is to develop an App that will control the robot using the tilt sensor in the mobile phone or tablet.

Activity	Linked	Skill Level	Intermediate

This worksheet describes how to use the outputs from an accelerometer to control the robot, so it's only really feasible to use it with a (mobile) device that has this capability.

Most mobile phones or tablets come with a 2 or 3 axis accelerometer (already built-in) that can detect when you tilt your device from side to side or slope it forwards or backwards. These are known as the X and Y axis. The third axis, the Z axis, shows movement when the device is raised or lowered. What the device is actually measuring is the rate of change of movement which is called acceleration.

This program makes use of the X and Y acceleration values to control the robot. For example, when you tilt your mobile device from side to the side the robot moves to the right or the left. When you tilt it forward or backwards this controls the speed and direction of the motors. Tilting forward increases the forward speed, and tilting backwards increases the speed in the reverse direction. Holding the mobile device level will stop the motors.

Accelerometer characteristics

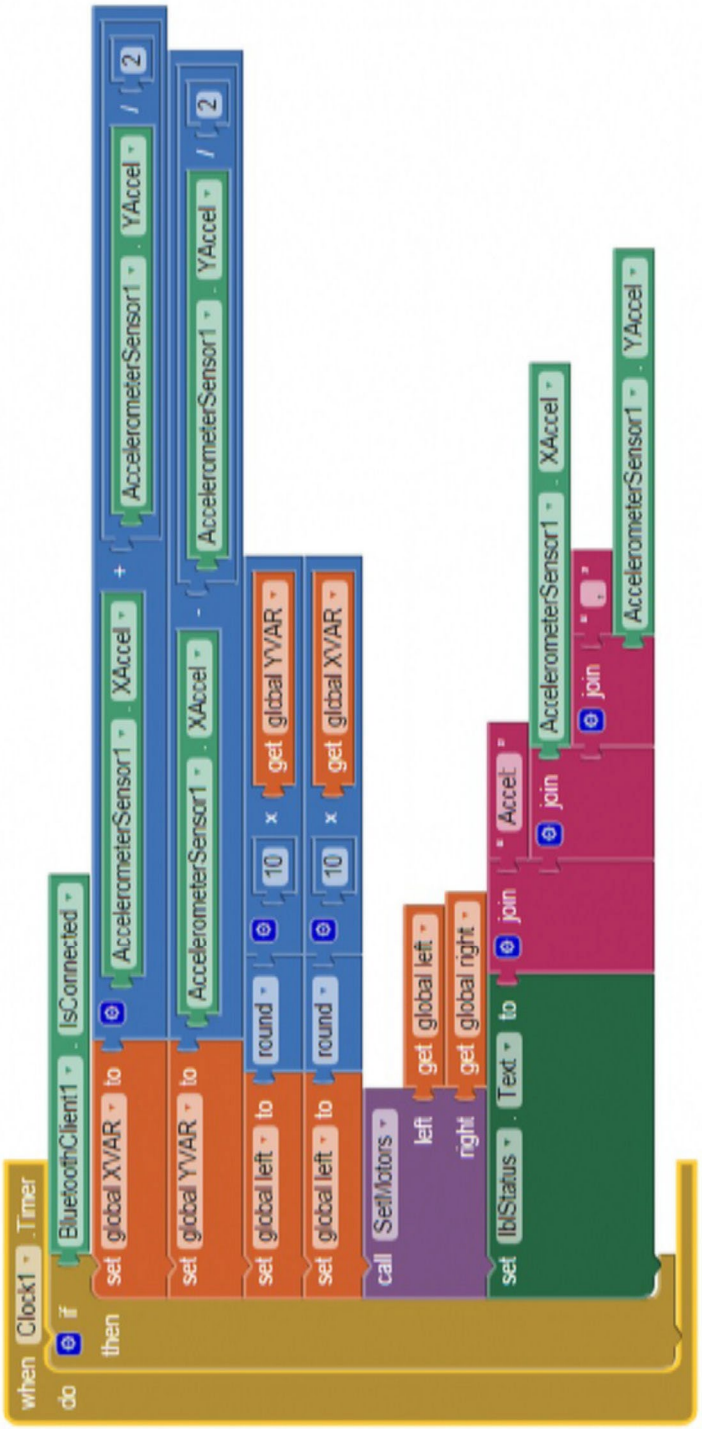
The X output from the accelerometer (let's name it xAccel) will give a value of 0 when the device is at rest on a flat surface, a positive value when it is tilted to the right (i.e. its left side is raised), and a negative value when it is tilted to the left (i.e. its right size is raised). The Y output (let's name it yAccel) will give a value of 0 when the device is at rest on a flat surface, positive when its bottom is raised, and negative when its top is raised.

Speed and Direction calculations

Once the X and Y acceleration values have been obtained they need to be converted into speed and direction values for the robot's two motors. You can do some research on the internet and investigate different methods of performing this calculation. However, to keep things simple, at this stage, the method used here is based on finding the average of the sum and difference of the X and Y values.

Shown below is an App Inventor program that applies the averaging technique and uses the resultant values as parameters for the API call to `SetMotors <left> <right>.`

Copyright © 2016 - 2026 Matrix Technology Solutions Limited



Over to you:

Once you have tried out this program you can have loads of fun driving the robot around remotely from a mobile device. Why not set up an obstacle course to navigate?

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

WORKSHEETS

Worksheet 11 - Lefty can Navigate Through a Maze



This worksheet brings together a number of tasks covered in previous sections of the guide book.

The objective is to develop a program that enables the robot to navigate a simple maze using the left-hand wall-following technique.

Are you prepared for the challenge ahead of you?

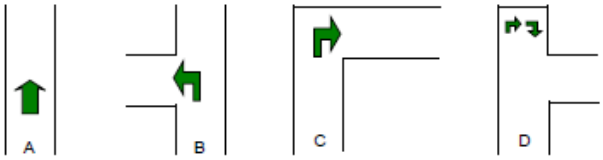
Activity	Linked	Skill Level	Intermediate

This exercise is probably one of the most challenging and rewarding things you can do with the robot. Making it move around and navigate a maze without it being touched appeals to most people regardless of their age. It also opens up the possibility of engaging in competitive challenges with friends to see who has the most agile and cleverest robot.

If you do some research on the Internet you will discover some of the many different methods people have developed for escaping from the centre of a maze or navigating through it. The most common method is the left-hand wall-following algorithm. An algorithm is a procedure or formula for solving a problem. It can be written informally as words or statements or in a structured way like a piece of pseudo code or a formal programming language.

Let's start off by assuming the robot has been placed at the entrance to a maze and see how a workable algorithm could be developed. At this stage it might be a good idea to share this exercise with a friend as you can bounce suggestions off of each other.

The robot has a set of distance sensors (positioned at the four corners and four edges) that can be used to detect if any obstacles are in the way. What the robot needs to do is check the left-hand distance sensor for the presence of the wall. It also needs to detect the distance sensor at the front of the robot to make sure the forward path is clear, before moving forwards. Situation (A) as shown in the diagram below.



If the wall (on the left-hand side) disappears, indicating a gap in the maze, then the robot needs to rotate to the left and continue following the wall. Situation (B) in the diagram.

If the robot detects its forward path is blocked, as in situation (C), then it needs to stop and rotate clockwise by 90 degrees, and then reassess the situation. In this situation the robot can proceed forward.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

If the robot has arrived at a dead-end, as in situation (D) overleaf, then it needs to stop and rotate clockwise by 90 degrees. When the robot reassesses the situation it will find that its forward path is still blocked, so it needs to rotate clockwise again by 90 degrees. The robot has effectively rotated by 180 degrees and can retrace its steps out of the deadend.

One way of looking at this is, to solve situation (D) you apply situation (C) twice. The algorithm is nearly complete, all that needs to be done is encapsulate it in a loop so the robot repeats the above steps on a cyclic basis. This solution is really a control program, just like the programs that are used to control industrial processes.

Here's an example of the program structure for navigating a maze. To assist you in reading through the program the API commands have been highlighted in blue.

As mentioned previously, if the robot encounters a dead-end the program will apply the rule for situation (C) twice in consecutive loops through the program.

```
// An example written in pseudo-code
// Navigate a maze using distance sensors DS-0 and DS-2

   clear = 100 //Path ahead is clear or there is no wall
   wall = 10 //Wall detected

   LOOP-FOREVER
   left = ReadIR 0 //Read distance sensor DS-0 (left-hand sensor)
   front = ReadIR 2 //Read distance sensor DS-2 (forward sensor)
   IF left <= wall AND front == clear THEN //Situation A
      SetMotors 50 50 //See note below
   ELSE-IF left == clear THEN //Situation B
      SetMotors 0 0
      Left 90
   ELSE-IF left <= wall AND front <= wall THEN //Situation C
      SetMotors 0 0
      Right 90
   END-IF
END-LOOP
```

Over to you:

Create the program described above to gain experience of using the distance sensors to control the speed and direction of the two motors.

Once you have managed to get your robot to navigate around a maze you should:
(a) Try changing the `SetMotors` values to see how fast you can get the robot to move around the maze without crashing into a wall.

(b) See if you can work out other control algorithms for maze navigation. For example, you could make use of the distance sensor on the right-hand side of the robot.

(c) See if you can make use of the microSD card to record the robot's journey.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

WORKSHEETS

Worksheet 12 - Play That Tune



This worksheet explains how to use the on-board loudspeaker and digital signal processor to play musical notes.

The second part of this exercise shows how to make a simple music interpreter so that you can transcribe sheet music and have fun getting the robot to play sounds.

Activity	Standalone	Skill Level	Easy

The robot can create digital audio tones that are played using the small on-board speaker. The frequency of the tones can range from 1 to 10,000 Hz and can have a duration from 1 to 10,000 ms. This means the robot covers a good portion of the audio range that humans can hear. To do this you would use the following API command.

```
PlayNote <note> <time>
```

For example to play middle C, which has a frequency of 262 Hz, for 1 second you use:

```
PlayNote 262 1000
```

To play a series of notes you just repeat the API call and substitute appropriate values. If you wanted to play a piece of music then this method would quickly become tiresome, so here's a method of making a very simple music interpreter. It is based on using letters to represent the musical notes and numbers to signify the duration of each note.

Starting at middle C the letter sequence would be C, D, E, F, G, A, B. Number 1 would mean one time-unit, 2 double that time, etc. You can decide how many milliseconds each time-unit represents. At this stage fractional notes, sharps and flats will be ignored, and only the first octave will be covered. You can extend the interpreter's capabilities later.

The first step is to find out the frequency for the notes in the octave starting from middle C. Middle C is 262 Hz, D is 294 Hz. You can find out the frequencies for the other notes.

Music Interpreter
To play a piece of music you need to take a page of sheet music and work out the name and duration of each note. For example "D" 2 means note D played for twice the time.

Then you can enter the details into the program on the next page.

The program consists of a macro that decodes the name of each note and its duration. So for example note "D" is replaced with the frequency value of 294 Hz. In a similar way the numeric values are replaced with a number that defines how many milliseconds to play the note. Once these two substitutions have been made an API call is made to PlayNote.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

```
// An example written in pseudo-code
// Music Interpreter
play(mynote duration)                                    //Macro to decode your music

     IF mynote == "C" THEN note = 262
     ELSE-IF mynote == "D" THEN note = 294
     ELSE-IF mynote == "E" THEN note = ???
     ELSE-IF mynote == "F" THEN note = ???
     ELSE-IF mynote == "G" THEN note = ???
     ELSE-IF mynote == "A" THEN note = ???
     ELSE-IF mynote == "B" THEN note = ???
     END-IF

     IF duration == 1 THEN time = 200                    //Time unit is 200ms
     ELSE-IF duration == 2 THEN time = 400
     ELSE-IF duration == 3 THEN time = 600
     ELSE-IF duration == 4 THEN time = 800
     END-IF

     PlayNote note time

END-play

CALL play("A" 1) //Can you work out the title of this music?

CALL play("G" 1)
CALL play("A" 1)
CALL play("C" 1)
CALL play("A" 1)

CALL play("G" 1)
CALL play("A" 1)
CALL play("C" 1)
CALL play("C" 1)

CALL play("D" 1)
CALL play("E" 1)
CALL play("D" 1)
CALL play("E" 1)

CALL play("D" 3)
CALL play("E" 1)
```

Over to you:

(a) Although this is a very simple music interpreter it lends itself to being extended and enhanced. First complete the program for the notes listed.

(b) Next, add extra decoding to handle 'sharps', 'flats', 'rests' and other octaves.

(c) Finally add some code to handle 'tempo'. So rather than having a basic time of 200 ms you could use tempo as a scaling factor to make the music sound more realistic.

This section has explained the simple `PlayNote` command and then shown that with a bit of ingenuity you can make a simple music player.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

WORKSHEETS

Worksheet 13 - Robo-DJ



This worksheet explains how to use the on-board loudspeaker and microSD card to play prerecorded material.

The second part of the exercise shows how you could build a simple sound playback machine.

Activity	Standalone	Skill Level	Easy

This simple exercise involves copying a piece of your favourite music onto a microSD card and then getting the robot to play it back to you. Pretty easy. Great fun.

The first step is to locate a pre-formatted microSD card. These are the tiny memory cards that fit into Smart phones and tablets. Next step is to temporally insert the card into your mobile device and copy a piece of music onto it. If your music is located on a desktop or laptop then you might need a SD-card adapter (unless your equipment is very new).

The recording format supported by the robot is Waveform Audio File Format commonly known as WAV because of its file extension. Ensure your sound copying package is set to output to the .wav format using one of the following sample-rate/bit-depth settings.

8KHz @ 8-bit, 8KHz @ 16-bit, 16KHz @ 8-bit or 16KHz @ 16-bit

Make sure the robot is switched OFF, then remove the microSD card from your copying equipment and insert it into the robot's microSD card slot. You're now ready to get the robot to play some music using the following API commands.

```
CardInit and CardPlayback <filename>
```

The first command initialises the robot's on-board electronics that control the SD card. The parameter <filename> defines the name of the .wav file to be played. This command actually returns a value (0 means OK, 239 means file not found, and 255 means error).

Here's an example to play a pre-recorded 'wav' file.

```
status = CardPlayback track1.wav
```

In a full length program you could arrange for a test to be made to see if the named variable status has a value other than zero, and if so flag an error accordingly.

Over to you:

You are now at the stage where you could record a series of 'wav' tracks on a microSD card and then write a really simple program to playback a single track through the robot's on-board speaker. For use later, label your sound tracks as track1.wav, track2.wav, etc. The next part of this exercise to develop a program to make a sound playback machine.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Sound playback machine

The example below shows a method of selecting tracks using SW1 to cycle round the available tracks and SW2 to instigate playing the selected track. If you plan to use a mobile device to perform these tasks you can use soft buttons on the device's screen.

```
// An example written in pseudo-code
  SW1_prev = 0
  SW2_prev = 0
  track = 1
  LCDPrint 0 0 "Track = "
  LCDNumber 54 0 track
  CardInit

  LOOP-FOREVER

   SW1 = ReadSwitch 0 // This is the track selector button
   IF SW1 > SW1_prev THEN // Detect rising-edge
      SW1_prev = SW1
      IF track < 10 THEN
         track = track +1
         LCDNumber 54 0 track
      ELSE track = 1
      END-IF
   ELSE-IF SW1 < SW1_prev THEN // Detect falling-edge
      SW1_prev = SW1
   END-IF

   SW2 = ReadSwitch 1 // This is the execute button
   IF SW2 > SW2_prev THEN // Detect rising-edge
      SW2_prev = SW2
      IF track == 1 THEN
         CardPlayback track1.wav
      ELSE-IF track == 2 THEN
         CardPlayback track2.wav
      ELSE-IF track == 3 THEN
         CardPlayback track3.wav
      ELSE-IF track == 4 THEN
         CardPlayback track4.wav
      etcetera
      END-IF
   ELSE-IF SW2 < SW2_prev THEN // Detect falling-edge
      SW2_prev = SW2
   END-IF
END-LOOP
```

Over to you:

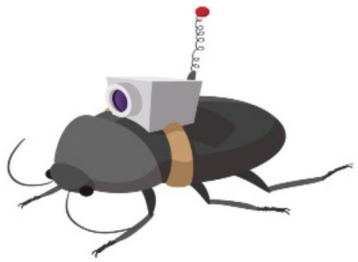
Although this is a very simple sound playback machine it lends itself to being extended and enhanced. Add extra decoding to handle 'forwards' and 'backwards' track selection rather than having to cycle round the tracks in one direction.

Now you know how to write a program to playback material, previously recorded on the microSD card, you will find it easier to tackle other worksheets in this Instructional Guide.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

WORKSHEETS

Worksheet 14 - Mobile Bug



This worksheet brings together some tasks that have been introduced in previous sections, to solve a typical real-world problem.

The problem that needs to be solved is to create a spying device. The robot drives to a location, records some speech, drives back and then plays back the recorded speech. It should be a lot of fun.

Activity	Linked	Skill Level	Intermediate

The objective for this exercise is to develop an App that will control the robot using the tilt sensor and also operate the sound facility to record and playback speech. Although this example uses App Inventor an equivalent package could be used to match your development and target platforms.

Don't be put-off. This exercise sounds more difficult than it really is. If you analyse the problem you will soon discover that you have already tackled some of the tasks. For example, the Tilt and turn worksheet explained how to drive the robot using a mobile device. You also found out how to operate the sound playback facility in the Robo-DJ worksheet, so the only item missing from the solution is learning how to record sound.

Sound recording using the robot is a little bit more involved, but not difficult, as the following API calls are available to make things fairly straight forward for you.

The first thing you need to do is initialise the robot's on-board electronics that control the microSD card. This task is performed with the API call `CardInit` . This call returns a value indicating the status of the system. Numeral 0 indicates system OK, 254 means an error, and 255 indicates that a card has not been detected in the microSD card slot.

The next step is to start recording from the on-board microphone to the microSD card.

The following API command is used to perform this task.

```
CardRecordMic <bitdepth> <samplerate> <time> <filename>
```

The <bitdepth> and <samplerate> parameters need to be set to match one of the robot's four supported recording modes. These are: 8-bit@8KHz, 16-bit@8KHz, 8-bit@16KHz or 16-bit@16KHz. The first setting gives the smallest file size on the SD card, while the fourth setting produces the best quality recording (but occupies a larger file size). Rather than entering the actual values a numeric code is used, as per the tables below.

	8KHz	16KHz
samplerate	0	1

	8-bit	16-bit
bitdepth	0	1

The <time> parameter defines the time-length of the recording. It can take a value from 1 to 65535 seconds. The last parameter specifies the name of the recording.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Here's an example to make a recording in the .wav format, at best quality, for 10 seconds.

```
CardRecordMic 1 1 10 track1.wav
```

This API call returns a value to indicate the status of the command. For example, it will return numeral 0 to indicate system OK, 1 indicates the file already exists on the card, and 255 means an error has occurred.

If you want to delete a recording from the card then you can use the following command.

```
CardDelete <filename>
```

The next step is create a user interface for the mobile device.

Here's an example of a simple layout as it would appear in the Designer view of App Inventor.



The first two soft buttons are concerned with the robot's Bluetooth communication facility.

The Stop button halts the movement of the robot by setting the motors' speed to zero.

The Start recording button issues the API command CardRecordMic (as detailed above).

The Erase recording button deletes the recording from the card. At the same time you could display a message on the mobile device or the robot's gLCD that shows...

"This recording will self-destruct after 15 seconds."

To help you get started with block-coding, there are some examples, on the next page, showing the events that are triggered in App Inventor's Block view.

Over to you:

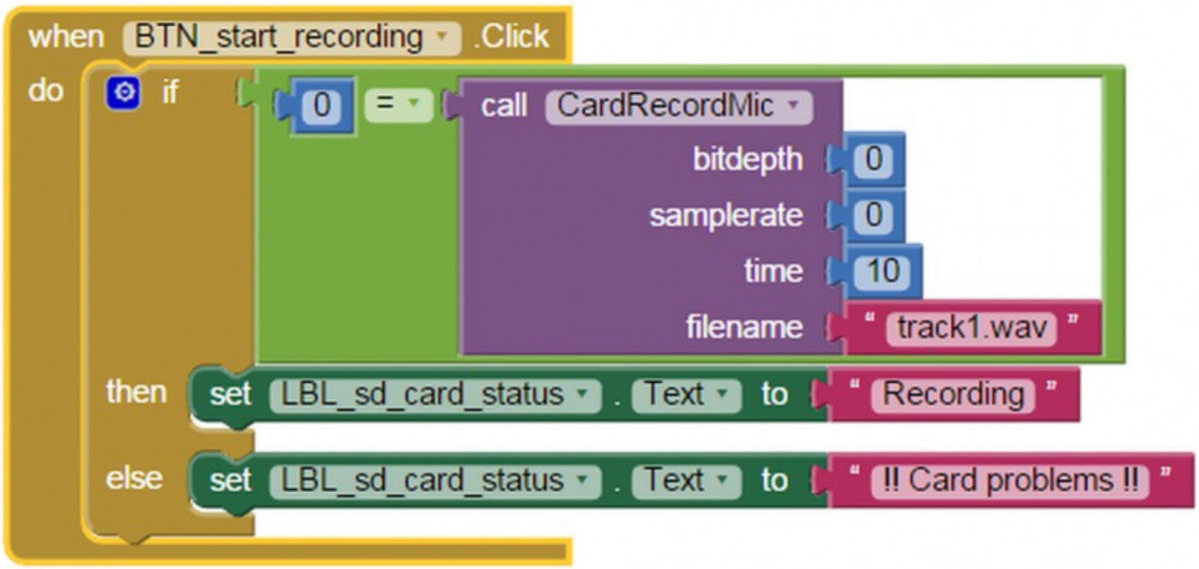
You should modify the Tilt and turn app to become Mobile bug. Once you've done that you could think about adding some extra functionality.

For example, write some code so that each time the Start recording button is pressed it records to a different filename. E.g. track1.wav, then track2.wav, etc.

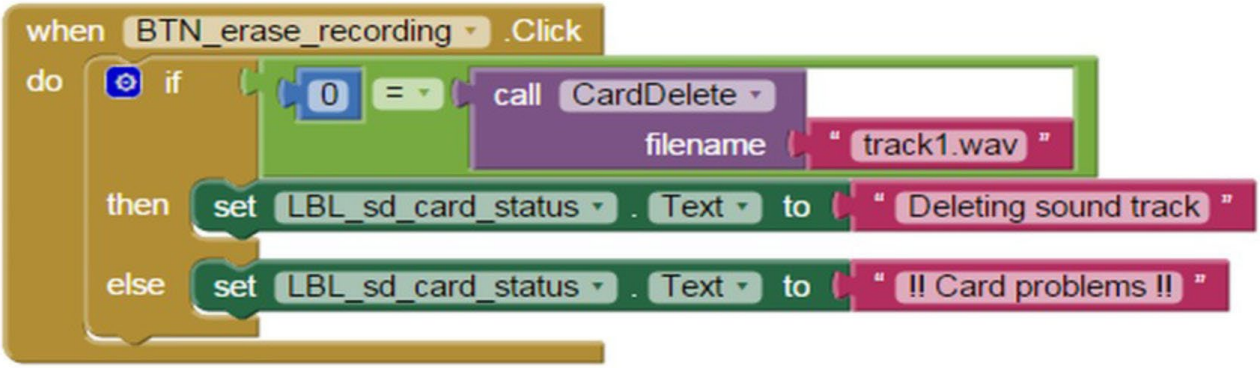
Another useful facility would be some sort of interlock to prevent a recording from being played until the robot is safely back at base: extend your program to arrange for a certain code to be entered before the playback facility can be activated.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Screen-shot of the events triggered by pressing the Start recording button.



Screen-shot of the events triggered when the !! Erase recording !! button is pressed.



Copyright © 2016 - 2026 Matrix Technology Solutions Limited

WORKSHEETS

Worksheet 15 - Robo-Pop



This worksheet brings together some tasks, covered in previous sections, to get the robot to dance to music when a sound (like clapping of hands) is detected by the microphone.

Instead of dancing you could program the robot to perform a 'celebration' as if it was a footballer scoring a goal. Liberate your imagination.

Activity	Linked	Skill Level	Intermediate

Don't be put-off. This exercise sounds more difficult than it really is. If you analyse the problem you will soon discover that you have already tackled most of the tasks. For example, the Let's move it, move it worksheet explained how to control the robot's motors. You also found out how to operate the sound playback facility in the Robo-DJ worksheet, so the only item missing from the solution is learning how to detect a sound.

The API call to detect a sound is **ReadMic**. This call returns a value so the way it would be used is to assign it to a named variable (e.g. sound_level) as shown below.

```
sound_level = ReadMic
```

The returned value can range between 0 to 4095 indicating the amplitude of the sound. You may need to carry out some experiments to determine the threshold-value to match the 'sound' you are planning on using as a 'trigger'.

This example shows how to combine a sound detector with other work you have done.

```
// An example written in pseudo-code
// Robo-pop triggered by sound detection

threshold = 100                          //Set this according to your situation

DO
sound_level = ReadMic                    //Clap your hands to make a noise
UNTIL sound_level >= threshold

CALL play_some_dance_music               //Use the code from "Robo-DJ"
CALL perform_some_groovy_moves           //Use the code from "Let's move it"
```

Over to you:

Once you've got the trigger-level set-up you can turn your attention to having some fun. Using the Let's move it, move it worksheet as a starting point you should develop some crazy moves. Likewise, using information from the Robo-DJ section you could include a soundtrack to complement the robot's moves. Try getting the robot to knock a fluffy ball into a makeshift goal and then perform its 'celebration piece' just like a footballer scoring a goal.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

WORKSHEETS

Worksheet 15 - Robo-Pop

You now have your robot dancing and playing some music. So what could be missing? Some lights, some laser beams - something to thrill the audience. Although the robot doesn't have any laser beams, it does have an LCD panel that can display graphical shapes - lines, rectangles or squares and manipulate individual pixels. Using the robot's ability to draw some rectangles you can make a do-it-yourself (DIY) light-beam show.

DIY light-beam show

The following API command instructs the graphical LCD panel to draw a rectangle.

```
LCDRect <x1> <y1> <x2> <y2>
```

The parameters <x1> and <y1> define the top-left of the rectangle and <x2> and <y2> bottom right. Here's an example to draw a square 10 by 10 pixels at coordinate 0,0.

```
LCDRect 0 0 10 10
```

Over to you:

Now that you've seen how to draw a black square, let's put a bit of animation into it. Your challenge is to make one or both of the black squares, shown in the diagram below, move across the LCD panel and back again (like a light beam on the dance floor at a disco).



If you make the starting point for square A at coordinate 0,4 and square B at 116, 18 then both squares will have space to slide over each other.

The next thing you need to consider is how to make the squares move.

The simplest method is to use a couple of named variables that act as counters to give the offset for the position of each square, and use them within the call to `LCDRect` .

```
LCDRect counterA 0 bottom_right_A 13
```

For example, the counter for square A starts at 0. To move the square to the right you increment the counter (by a suitable value, say 5) and call `LCDRect` again. Bottom_right is another named variable that takes the value: bottom_right_A = counterA + 10.

You will need to insert a short delay before the sequence is repeated so that the black square moves smoothly across the screen. Your program you will need to check when counterA reaches the far-right of the LCD panel - and decide whether to reset it to its starting point, or just retrace its movement (as in the diagram).

The animated movement for square B can be implemented in a similar way as square A using two more named variables (e.g. counterB and bottom_right_B).

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

TAKING IT FURTHER

Challenges 16-20



The final 5 challenges are collected here for those who want to take things further.

We are not giving you much guidance here because the majority of work here builds on the previous worksheets and if you've got this far you'll be a programming expert!

Challenge 16 - Logging data
There are lots of possibilities here: Use the SD card API commands to log the sensor data on a journey or the light level as the sun goes down in the evening. You could also spin whilst logging the light level to produce a 360° graph of light in the room.

Challenge 17 - Using the servos
Attach servo motors to the expansion port and turn the robot into a digger or give it a "foot" to kick a ball with.

There are 4 servo motor attachments at the front of the robot and the commands to control them are listed in the API reference.

Challenge 18 - Pimp my ride
Design a shell for the Formula AllCode robot which makes your robot stand out. Perhaps you can use a 3D printer to create a truly unique body for the robot.

Attach to the robot's chassis using the 4 holes near the wheels.

Challenge 19 - Customised electronics
Extend the individuality of your robot by adding LEDs and other custom electronics to its expansion ports.

Remember to wire your electronics correctly so you do not damage the microcontroller!

Challenge 20 - Acceleration and orientation
Did you know there is an inbuilt accelerometer? You can use this to measure how much the AllCode robot is accelerating, what forces exist when it turns a corner, and even which way up it is (Einstein taught us that gravity and acceleration are essentially the same thing!).

The API command for reading the Accelerometer is `ReadAxis`.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

APPENDIX 1
API Reference Chart

This section lists the API calls by their functional groups so it's easy to find the particular command you're looking for. There are different types of call - some return a value, some don't. Some require one or more parameters others don't.

For example, the API call **`SetMotors`** requires two parameters to define the speed of the two motors, whereas the call to **`ReadLight`** just returns a value (i.e. light sensor value). There are other calls that require a parameter and will also return a value.

A good example is the **`ReadIR`** API call. This call needs a parameter to define which IR sensor you want to check. The return value is the distance measurement for that particular sensor.

Connection			
Return	Command	Parameters(s)	Description
status	`ComOpen`	port	Open COM port Port= 1 to 255 Status= 0(OK) or 255(error)
status	`ComClose`		Close port Status= 0(OK) or 255(error)
Version	`GetAPIversion`		Returns the version number of the API Version+ 1 to 65535

Sensors			
Return	Command	Parameters(s)	Description
Value	`ReadSwitch`	index	Read the switch value Index= 0(left) or 1(right) Value= 0(false) or 1(true)
Value	`ReadIR`	index	Reads an IR sensor Index= 0 to 7 Value= 0 to 4095
Value	`ReadLine`	index	Reads a line sensor Index= 0(left) or 1(right)
Value	`ReadLight`		Reads a light sensor Value= 0 to 4095
Value	`ReadMic`		Reads microphone sensor Value= 1 to 4095
Value	`ReadAxis`	index	Reads an axis of the accelerometer Index= 0 (x), 1 (y) or 2 (z) Value= -32768 to 32768

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

APPENDIX 1
API Reference Chart

Motors			
Return	Command	Parameters(s)	Description
	`SetMotors`	Left Right	Set the speed of the motors Left = -100 to 100 Right = -100 to 100
	`Forwards`	distance	Move torwards distance (in mm) Distance = 0 to 1000
	`Backwards`	distance	Move backwards distance (in mm) Distance = 0 to 1000
	`Left`	angle	Turn left angle (in degrees) Angle = 0 to 360
	`Right`	angle	Turn right angle (in degrees) Angle = 0 to 360

LED/Speaker			
Return	Command	Parameters(s)	Description
	`LEDWrite`	value	Write value to the LEDs Value = 0 to 7
	`LEDOn`	index	Turn an LED on Index = 0 to 7
	`LEDOff`	index	Turn and LED off Index = 0 to 7
	`Playnote`	note time	Output audio note (in Hz) for time (in ms) Note = 1 to 1000 Time = 1 to 1000

Servo			
Return	Command	Parameters(s)	Description
	`ServoEnable`	index	Enable a servo channel Index = 0 to 3
	`ServoDisable`	index	Disable a servo channel Index = 0 to 3
	`ServoSetPos`	index position	Set a servo position Index = 0 to 3 Position = 0 to 255
	`ServoAutoMove`	index position	Auto-move to a servo position Index = 0 to 3 Position = 0 to 255
	`ServoMoveSpeed`	speed	Set servo auto-move speed Speed = 1 to 50

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

APPENDIX 1
API Reference Chart

LCD			
Return	Command	Parameters(s)	Description
	`LCDClear`		
	`LCDPrint`	X Y text	Print text on the LCD X = 0 to 127 Y = 0 to 31 Text = <string>
	`LCDNumber`	X Y value	Print an integer value on the LCD X = 0 to 127 Y = 0 to 31 Value = -32768 to 32767
	`LCDPixel`	X Y state	Draw a pixel on the LCD X = 0 to 127 Y = 0 to 31 State = 0 (off) or 1 (on)
	`LCDLine`	X1 Y1 X2 Y2	Draw a line on the LCD X1= 0 to 127 Y1 = 0 to 31 X2 = 0 to 127 Y2 = 0 to 31
	`LCDRect`	X1 Y1 X2 Y2	Draw a rectangle on the LCD X1 = 0 to 127 Y1 = 0 to 31 X2 = 0 to 127 Y2 = 0 o 31
	`LCDBacklight`	Value	Set the LCD backlight brightness Value = 0 to 100
	`LCDOptions`	Foreground Background Transparent	Sets option for drawing on the LCD Foreground = 0 (white) or 1 (black) Background = 0 (white) or 1 (black) Transparent = 0 (false) or 1 (true)

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

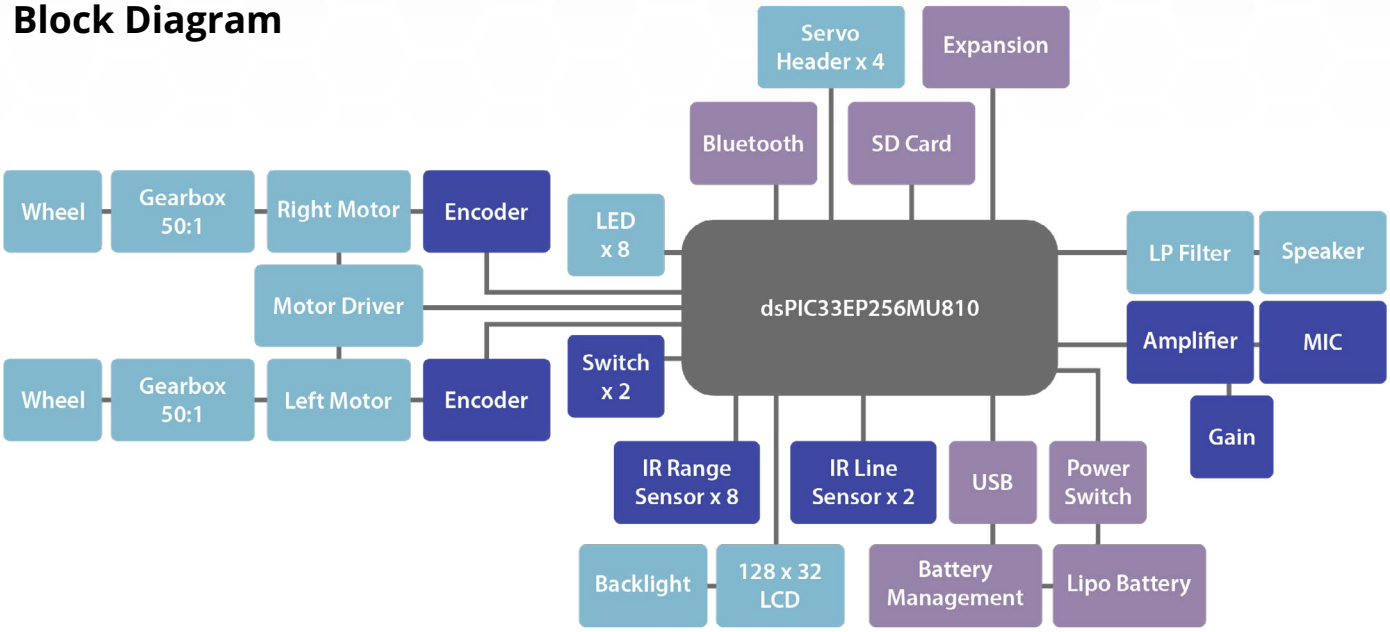
APPENDIX 1
API Reference Chart

SD Card			
Return	Command	Parameters(s)	Description
status	`CardInit`		Initialise the SD card Status = 0 (OK), 254 (error) or 255 (no card)
status	`CardCreate`	filename	Create new file Filename = <string> Status = 0 (OK), 1 (file exists) or 255 (error)
status	`Cardopen`	filename	Open an existing file Filename = <string> Status = 0 (OK), 239 (file not found) or 255 (error)
status	`CardDelete`	filename	Delete a file Filename = <string> Status = 0 (OK) or 255 (error)
status	`CardWriteByte`	data	Write a byte of data to the open file Data = 0 to 255 Status = 0 (OK) or 255 (error)
data	`CardReadByte`		Read a byte of data from the open file Data = 0 to 255
status	`CardRecordMic`	Bitdepth Samplerate Time filename	Record the microphone Bitdepth = 0 (8bit) or 1 (16bit) Samplerate = 0 (8K) or 1 (16K) Filename = <string> Status = 0 (OK), 239 (file exists) or 255 (error)
status	`CardPlayback`	filename	Play an audio file Filename = <string> Status = 0 (OK), 239 (file not found) or 255 (error)
status	`CardBitmap`	X Y filename	Display an image on the LCD X = 0 to 127 Y = 0 to 31 Filename = <string> Status = 0 (OK), 239 (file not found) or 255 (error)

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

APPENDIX 2
Microcontroller Pin Connections

Block Diagram



Microcontroller Connections

Motors	
Left Speed Control	D1
Left Direction Control	D0
Left Encoder Feedback	B11
Right Speed Control	D2
Right Direction Control	D3
Right Encoder Feedback	B12

LCD	
Data	G8
Clock	G6
A0	A3
CS	A2
Reset	A4

LEDs	
D0	E0
D1	E1
D2	E2
D3	E3
D4	E4
D5	E5
D6	E6
D7	E7

Bluetooth	
TX	F12
RX	F8
Pair Status	F0
Reset	F13

Servo Headers	
Servo1	A7
Servo2	A9
Servo3	A10
Servo4	A14

IR Range	
IR1 Enable	D8
IR1 Signal	B4
IR2 Enable	D11
IR2 Signal	B5
IR3 Enable	D12
IR3 Signal	B14
IR4 Enable	D13
IR4 Signal	B15
IR5 Enable	D15
IR5 Signal	B8
IR6 Enable	A5
IR6 Signal	B9
IR7 Enable	A6
IR7 Signal	B10
IR8 Enable	C4
IR8 Signal	E8

Extras	
Microphone	B3
Speaker	F3
Light Sensor	B2
Battery Voltage	E9

EBlocks Expansion	
1	E0
2	E1
3	E2
4	E3
5	E4
6	E5
7	E6
8	E7
9	GND
10	GND

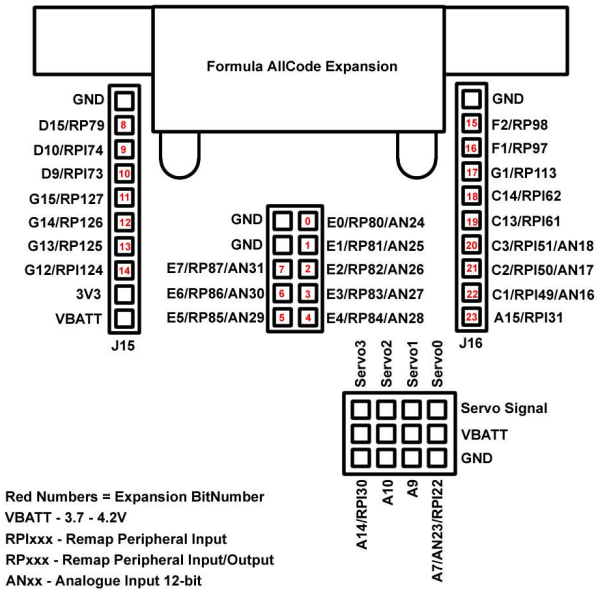
Expansion J15	
1	VBATT
2	3V3
3	G12
4	G13
5	G14
6	G15
7	D9
8	D10
9	D15
10	GND

Expansion J16	
1	A15
2	C1
3	C2
4	C3
5	C13
6	C14
7	G1
8	F1
9	F2
10	GND

SD Card	
MOSI	D4
MISO	D5
SCK	D6
CS	D7

USB	
USB Detect and LED	B13
D+	G2
D-	G3

Switches	
SW1	A0
SW2	A1



Copyright © 2016 - 2026 Matrix Technology Solutions Limited

APPENDIX 3
Musical Note Frequencies

Note	Freq (Hz)
C_1	33
$C^{\#}_1/D^{b}_1$	35
D^1	37
$D^{\#}_1/E^{b}_1$	39
E_1	41
F_1	44
$F^{\#}_1/G^{b}_1$	46
G^1	49
$G^{\#}_1/A^{b}_1$	52
A^1	55
$A^{\#}_1/B^{b}_1$	58
B^1	62

Note	Freq (Hz)
C_2	65
$C^{\#}_2/D^{b}_2$	69
D^2	73
$D^{\#}_2/E^{b}_2$	78
E_2	82
F_2	87
$F^{\#}_2/G^{b}_2$	93
G^2	98
$G^{\#}_2/A^{b}_2$	104
A^2	110
$A^{\#}_2/B^{b}_2$	117
B^2	123

Note	Freq (Hz)
C_3	131
$C^{\#}_3/D^{b}_3$	139
D^3	149
$D^{\#}_3/E^{b}_3$	456
E_3	165
F_3	175
$F^{\#}_3/G^{b}_3$	185
G^3	196
$G^{\#}_3/A^{b}_3$	208
A^3	220
$A^{\#}_3/B^{b}_3$	233
B^3	247

Note	Freq (Hz)
C_4	262
$C^{\#}_4/D^{b}_4$	277
D^4	294
$D^{\#}_4/E^{b}_4$	311
E_4	330
F_4	349
$F^{\#}_4/G^{b}_4$	370
G^4	392
$G^{\#}_4/A^{b}_4$	415
A^4	440
$A^{\#}_4/B^{b}_4$	466
B^4	494

Note	Freq (Hz)
C_5	523
$C^{\#}_5/D^{b}_5$	554
D^5	587
$D^{\#}_5/E^{b}_5$	622
E_5	659
F_5	698
$F^{\#}_5/G^{b}_5$	740
G^5	784
$G^{\#}_5/A^{b}_5$	831
A^5	880
$A^{\#}_5/B^{b}_5$	932
B^5	988

Note	Freq (Hz)
C_6	1047
$C^{\#}_6/D^{b}_6$	1109
D^6	1175
$D^{\#}_6/E^{b}_6$	1245
E_6	1319
F_6	1397
$F^{\#}_6/G^{b}_6$	1480
G^6	1568
$G^{\#}_6/A^{b}_6$	1661
A^6	1760
$A^{\#}_6/B^{b}_6$	1865
B^6	1976

Note	Freq (Hz)
C_7	2093
$C^{\#}_7/D^{b}_7$	2217
D^7	2349
$D^{\#}_7/E^{b}_7$	2489
E_7	2637
F_7	2794
$F^{\#}_7/G^{b}_7$	2960
G^7	3136
$G^{\#}_7/A^{b}_7$	3322
A^1	3520
$A^{\#}_7/B^{b}_7$	3729
B^7	3951

Note	Freq (Hz)
C_8	4186
$C^{\#}_8/D^{b}_8$	4435
D^8	4699
$D^{\#}_8/E^{b}_8$	4978
E_8	5274
F_8	5588
$F^{\#}_8/G^{b}_8$	5920
G^8	6272
$G^{\#}_8/A^{b}_8$	6645
A^8	7040
$A^{\#}_8/B^{b}_8$	7459
B^8	7902

Note	Freq (Hz)
C_9	9372
$C^{\#}_9/D^{b}_9$	8870
D^9	9398
$D^{\#}_9/E^{b}_9$	9956
E_9	10548
F_9	11176
$F^{\#}_9/G^{b}_9$	11840
G^9	12544
$G^{\#}_9/A^{b}_9$	13290
A^9	14080
$A^{\#}_9/B^{b}_9$	14918
B^9	15804

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

APPENDIX 4

Setting the Robot's Name

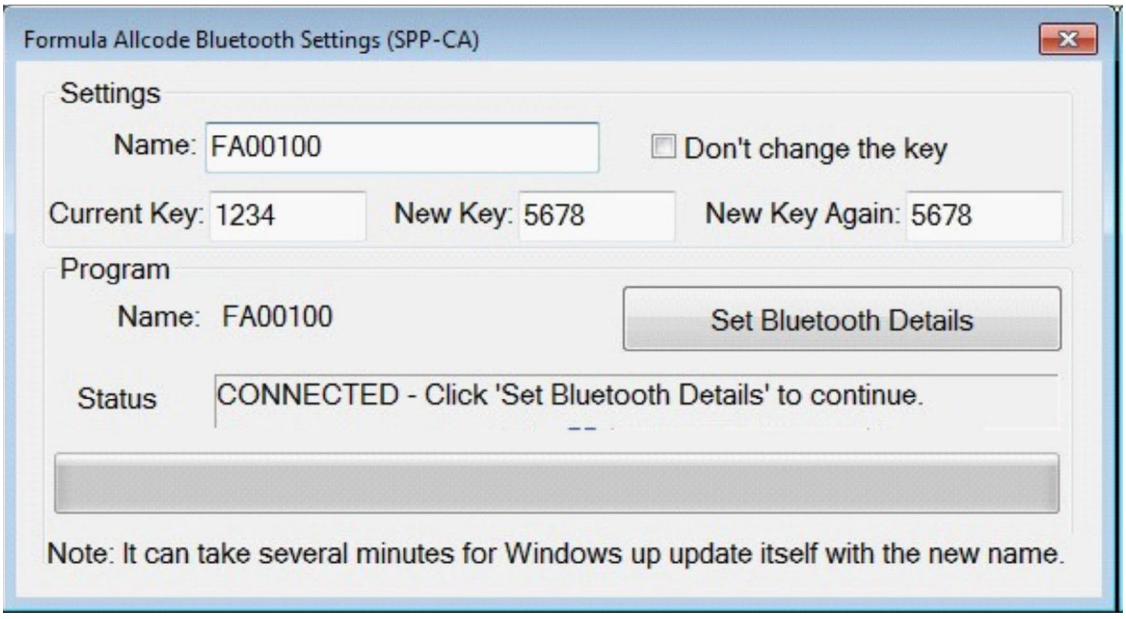
The Bluetooth name for the Formula AllCode can be changed by using a PC with the mLoader software installed. The mLoader software can be downloaded from the MatrixTSL website.

http://www.matrixtsl.com/formula-allcode/

Once you have downloaded the mLoader software there is a file in the software folder named Formula AllCode Rename. Double click this file to start the mLoader software in the Bluetooth rename mode.

Connect the Formula AllCode robot to the PC using the USB cable provided and press the reset button on the robot to allow the mLoader software to see the robot.

Once you have done this you can use the mLoader software to set the Bluetooth name and pair key (sometimes called the passcode) by entering the required details into the text fields and then using the Set Bluetooth Details button to transfer your settings to the robot.



We recommend you keep the key set to the default "1234" unless you are working in a group of other users and want to prevent others from connecting to your robot. If you do change the pass key, you will need to enter the existing key, so try to remember it!

If the key is changed and you cannot remember it, please contact Matrix for instructions on how to reset it.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

APPENDIX 5
Reloading the AllCode API Firmware

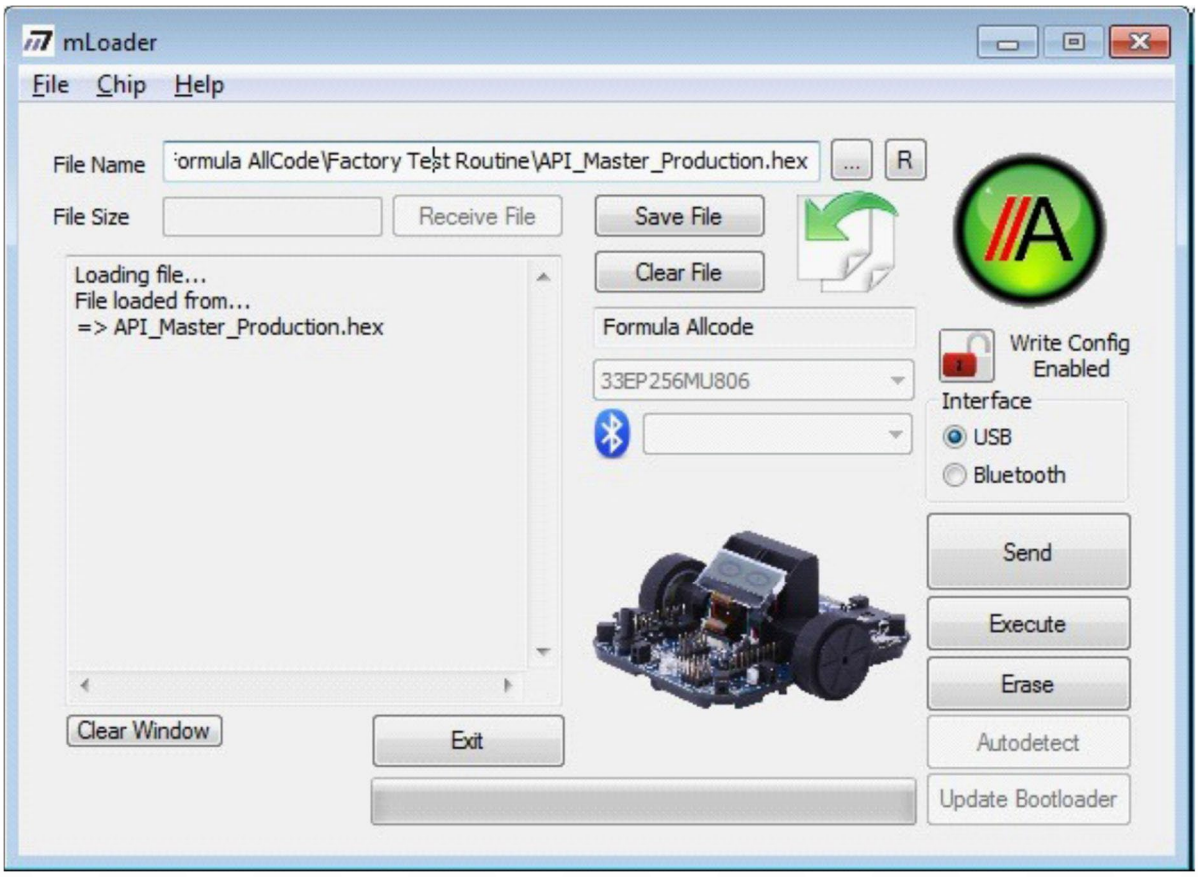
By default the Formula AllCode robot will arrive with the API firmware already installed. As the robot is reprogrammable using languages like Flowcode or C, it is possible to overwrite the factory AllCode API functionality. It is also possible that newer versions of the API firmware are released and you may want to upgrade to the latest version.

The API firmware is available to download in the form of a hex file from the Formula AllCode page of the MatrixTSL website. To load the hex file onto the Formula AllCode you need a copy of the latest mLoader software which is also available from the website:

https://www.matrixtsl.com/allcode/resources/

You can connect the Formula AllCode to the PC using the supplied USB cable or using the Bluetooth data connection. Open the mLoader software and press the reset button on the robot to allow the mLoader software to see the robot.

Click the "..." icon next to the File Name text field to select the hex file to send to the AllCode.



Click the Send button to transfer the hex file to the Formula AllCode and restore the API firmware. Remember to press the execute button once the firmware has been sent to allow the Formula AllCode firmware to run.

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Notices

Trademarks
PIC, PICMicro and dsPIC are registered trademarks of Microchip Technology. Raspberry Pi is a registered trademark of the Raspberry Pi Foundation.

Other product names that appear in this document may be trademarks of their own respective owners.

Easy

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Notes

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

Notes

Copyright © 2016 - 2026 Matrix Technology Solutions Limited

ROBOTICS COURSE - INSTRUCTIONAL GUIDE

CP5894

www.matrixtsl.com

Matrix Technology Solutions
The Factory, 33 Gibbet Street, Halifax, HX1 5BA, United Kingdom
t: +44 (0) 1422 252380 e: sales@matrixtsl.com