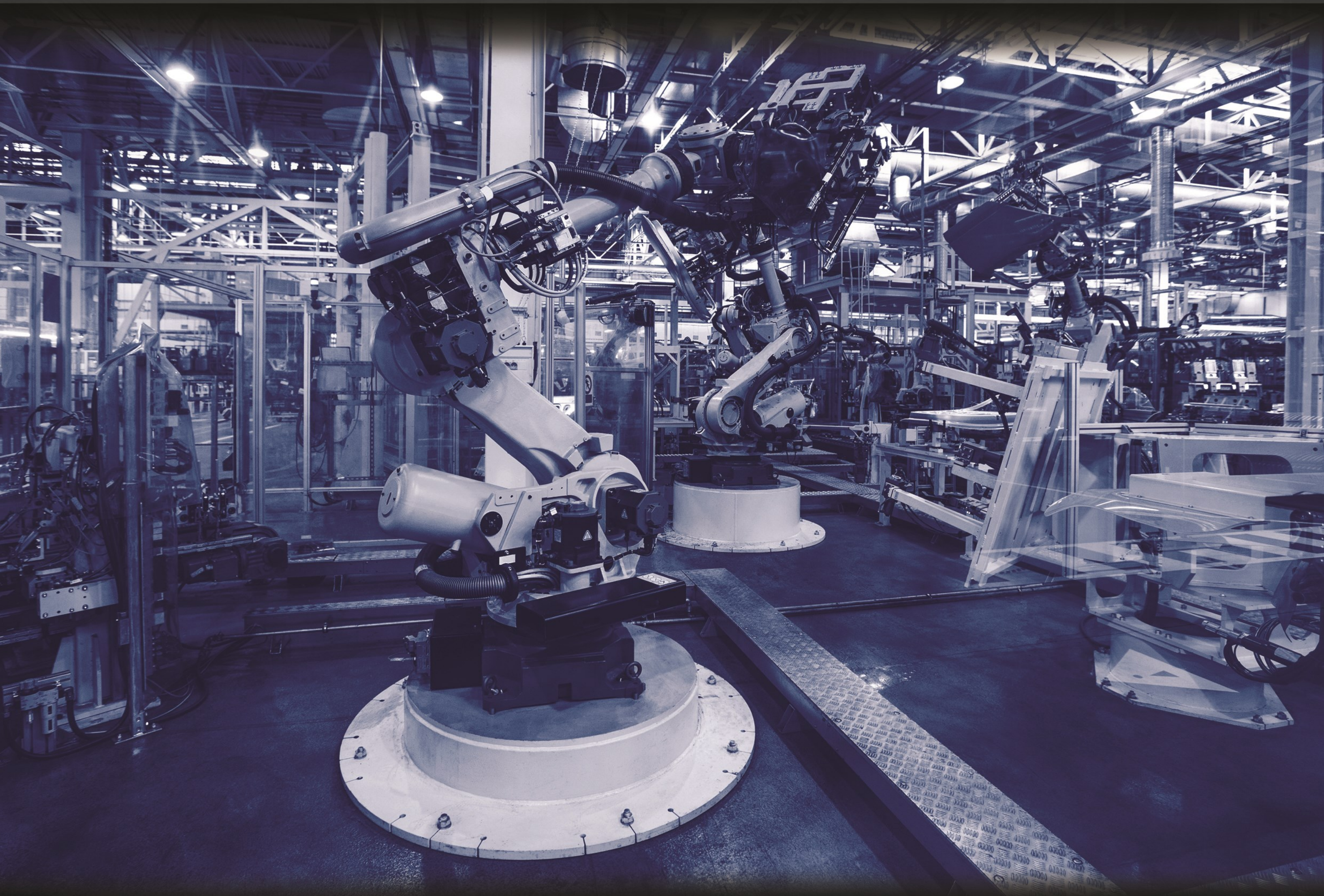
//ALLCODE
ROBOTARM
Production Cell

Robot arm development



CP8656

MATRIX
www.matrixtsl.com

Contents

Introduction

Getting Started

Where do you start?

Congratulations. You now have a state-of-the-art AllCode Robot arm that can perform all sorts of interesting manoeuvres, pick up objects, detect touch pressure, sense an object's colour and make intelligent decisions. 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 arm to make its first moves 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 arm 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 manually moving the arm, operating the LEDs or displaying a message on the LCD panel, then move on to more complicated tasks like picking, placing and sorting 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 arm. This will ensure that you are focused and productive with your time.

Powering up the robot arm

To power up the robot arm refer to the power supply instructions located at the end of page 7. There is a picture and short explanation of which power supply goes where.

Testing the robot arm

There are two in built demonstration modes in the supplied firmware which allow you to test out the robot arm before you start going through the exercises. Pages 27 and 28 show how these demo modes function and what to expect.

Pairing your Bluetooth enabled device to the robot arm

Pages 11 through 16 show how to pair the robot arm to various different devices. The pairing is required for exercises which use the API communications to control the robot arm.

The Pre-Programmed Firmware

The USB cable supplied is not needed unless you want to re-program the in-built firmware on the robot arm. The original firmware can be found on the AllCode section of the Matrix Website and can be re-programmed using the mLoader programming tool.

If you find you do want to re-program the in-built firmware then the USB cable needs to be connected to the USB port on the EB091 board marked **Ghost**.

Take the Tour

Features of the Robot Arm

This section explores the various parts of the robot arm and explains the basic operation of all the sensors and control systems.

Apart from the six motors that can be used to move and orient the various joints of the robot arm, there are colour and light sensors as well as switches, LEDs and a display that will give you hours of fun.

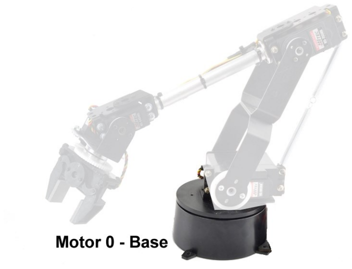
You will probably want to revisit this page as you read this Guide.

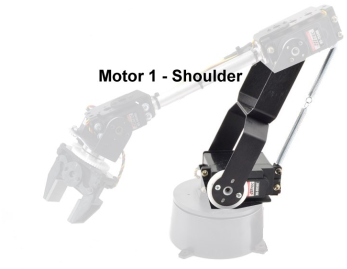
WARNING

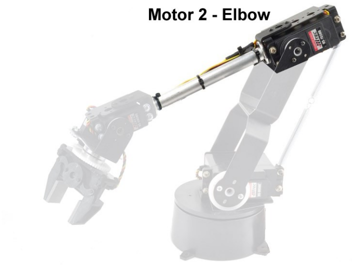
The robot arm is an active mechatronics system. To avoid injury please take care when operating the robot. Always ensure to place the robot into the parked position when switching off the robot or changing mode. Always ensure the arm is in the parked position when powering up to ensure no sudden high speed movement.

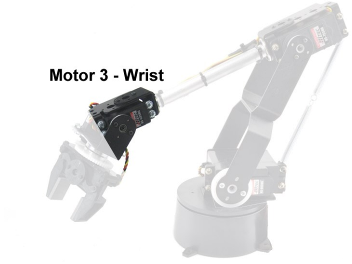
Key parts of the robot arm

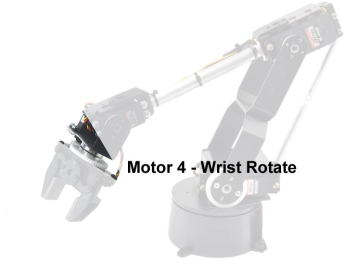
The images below shows the various moving parts of the AllCode Robot Arm.

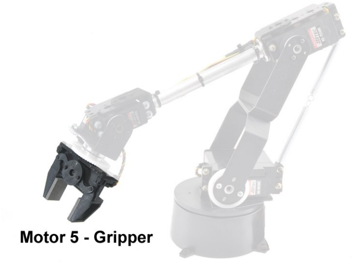






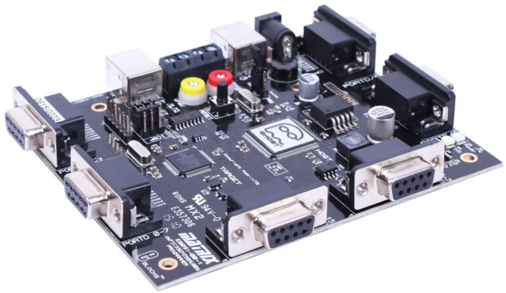






Take the Tour

Features of the Robot Arm



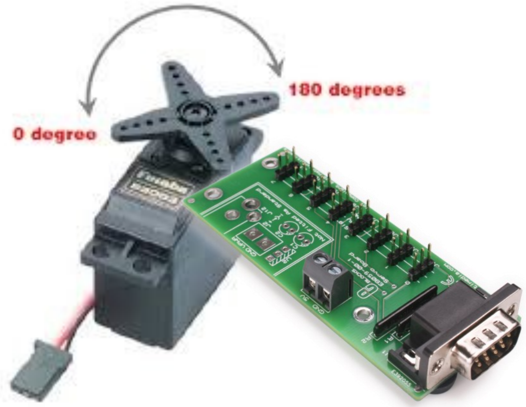
EB091 dsPIC Ghost Board

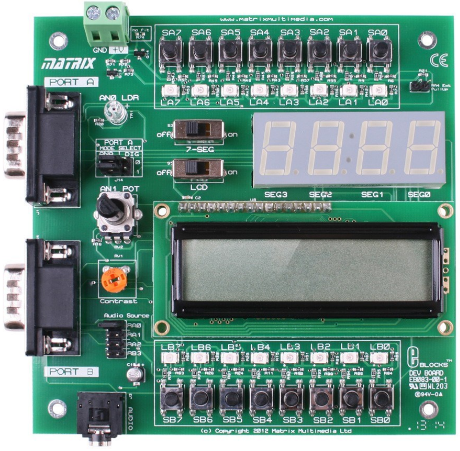
At the heart of the robot arm is a Digital Signal Controller (dsPIC®) manufactured by Microchip Technology Inc.

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.

Servo Motors

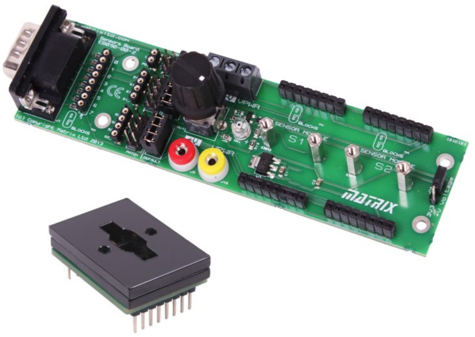
Each of the rotating joints on the robot arm are driven using a servo motor. These motors are capable of rotating to any fixed angle within a range of approx. 180 degrees. Inside the motor is a DC motor, a feedback mechanism and a controller which monitors the feedback and control signal and adjusts the motor speed and direction as required. Each servo motor connects to the EB091 Board via a EB059 Servo Board.





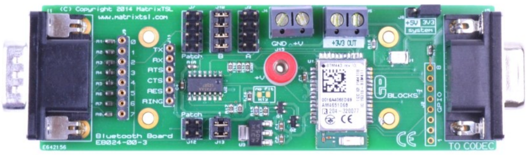
EB083 Combo Board

A way for the user to interface to the system is always very useful. The combo board features switches and LEDs which can be used to control the robot arm as well as a display that can show various status messages and joint angles.



EB090 Sensors Board

The sensors board allows a number of distinct sensors to be connected to the system to allow us to detect and measure various real world characteristics. In this case we have a colour sensor and a Buzzer/Touch feedback sensor.



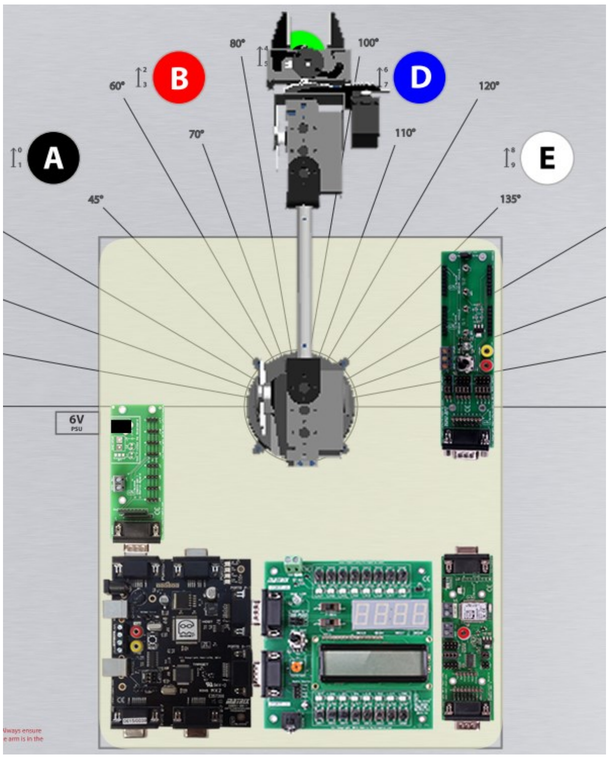
EB024 Bluetooth Board

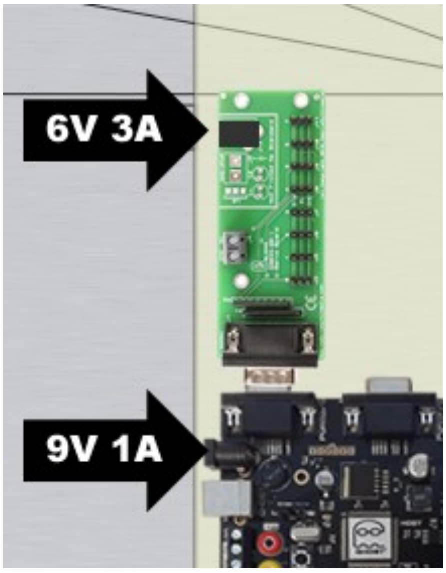
The Bluetooth board allows the robot arm to be controlled via Bluetooth enabled devices such as personal computers, single board computers, mobile phones and tablets.

Bluetooth capabilities can be added to a computer using a standard low cost Bluetooth USB dongle.

The Robot Arm Exercise Mat

The robot arm needs to be placed on to the exercise mat so that the plastic base plate is contained and in line with the place marker on the mat.





Powering the Robot Arm

The Robot Arm needs to be powered via the supplied 6V 3A DC Power Supply. The power supply is connected to the DC socket on the EB059 Servo E-block as indicated on the mat.

The EB091 and other E-blocks need to be powered via the supplied variable voltage DC Power Supply. Set this to output at 9V and connect to the DC socket on the EB091 board.

Language Neutral

Application Program Interface

Robotic Arm Course Instructional Guide

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 AllCode Robot Arm 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

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.

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 AllCode Robot Arm, offers the same platform and language independence as the TV example described above. The difference is the transmission medium for the robot is Bluetooth or Wifi rather than an infrared beam. This means providing you have a Bluetooth or Wifi 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, a Bluetooth enabled PC/Mac/Raspberry Pi® or a Matrix E-block upstream board would do the same by fitting a low-cost Bluetooth module to create a simple controller.

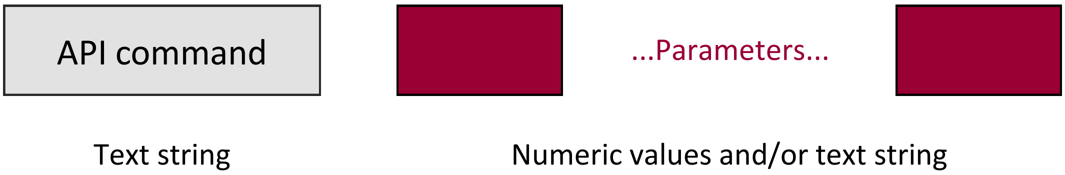
Language Neutral

Application Program Interface

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 is the colour sensor fitted to the robot arm panel. An API command could be sent to sample the sensor (that effectively interrogates it) causing it to return a numerical value of the colour of light reflected back from an object positioned over the sensor.

Shown below is the general format for the robot's API.



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 combo board 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 "`ra.LEDWrite(20)`" whereas C# would look like "`RA_DLL.RA_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.

```
SetServo <index> <position>
```

The parameter labelled <index> can take a value between 0 and 5 to define which motor to move. The parameter labelled <position> can take a value between 0 to 180 to define the angle the motor should move to.

If you wanted to control all motors at once you could use this command.

```
SetAllServos <position0> <position1> <position2>
<position3> <position4>
```

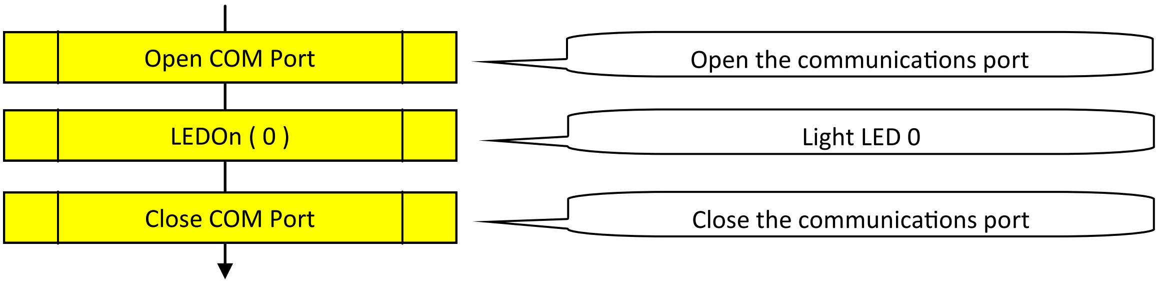
The parameters <positionX> can each take a value from 0 to 180 to define the motor angles in degrees.

Pseudo-code

A Simple Programming Aid

Most of the examples that appear in this Instructional Guide are written with pseudo-code. This pseudo-code can be used with the many programming languages that can be used with the AllCode Robot Arm.

One of the reasons for this is that 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.



API calls allow you to interact with the AllCode Robot Arm. The name of these calls will map across to the set of macros available in Flowcode, AppInventor, Python and other languages supported by the AllCode Robot Arm. A full list of available API calls is available at the end of this document.

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, which should hopefully be a fairly simple coding task.

Hardware setup

Windows PC

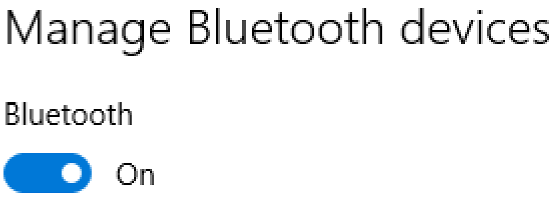
A lot of Windows devices, especially laptops and tablets, have inbuilt Bluetooth functionality. If your PC does not, you will need to use a Bluetooth 2 USB dongle.

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.



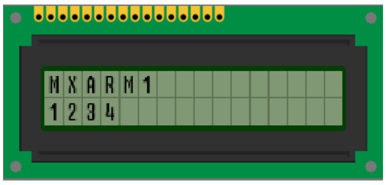
Manage Bluetooth devices

Bluetooth

On

2) Pair the robot

First switch on the robot - the Bluetooth device name and pair key will be displayed on the LCD screen for a few seconds. Pressing the reset button on the EB091 board allows you to see the details again.



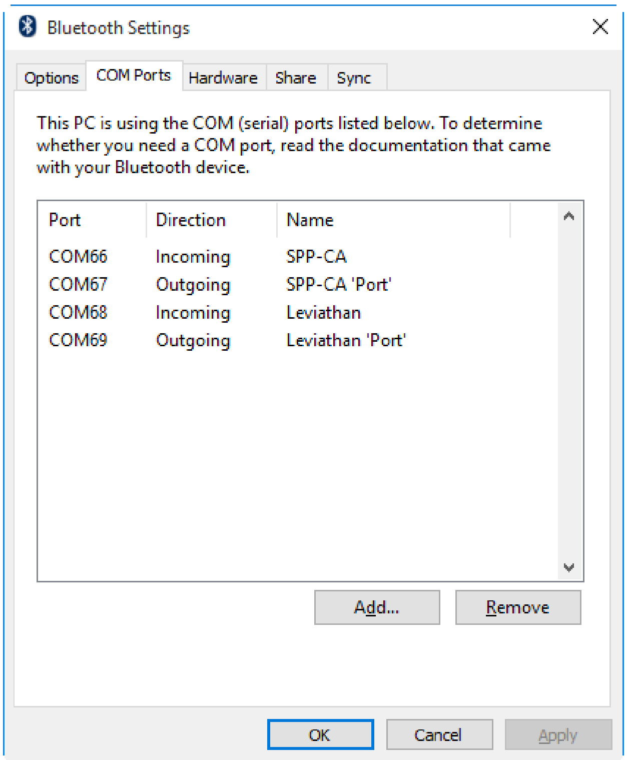
Again, pairing works slightly differently on the various Windows versions and so it is difficult to give specific instructions here. The Windows help and website will have guides explain how.

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 AllCode Robot Arm 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.

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.



This PC is using the COM (serial) ports listed below. To determine whether you need a COM port, read the documentation that came with your Bluetooth device.

Port	Direction	Name
COM66	Incoming	SPP-CA
COM67	Outgoing	SPP-CA 'Port'
COM68	Incoming	Leviathan
COM69	Outgoing	Leviathan 'Port'

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

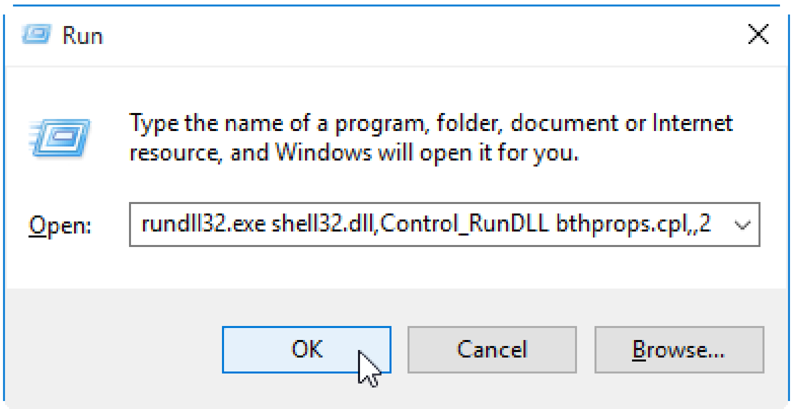
If you did not see the COM port when the robot was paired, you can find it in the Bluetooth Settings window, as shown on the right.

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
```



Type the name of a program, folder, document or Internet resource, and Windows will open it for you.

Open: 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 AllCode Robot Arm.

Hardware setup

Android Tablet / Phone

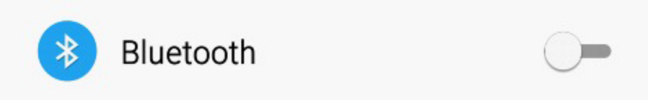
Android phones and tablets, when used with intuitive programming software such as App Inventor, provide a motivating platform for controlling the AllCode Robot Arm.

These devices almost always include Bluetooth and Wifi built-in.

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

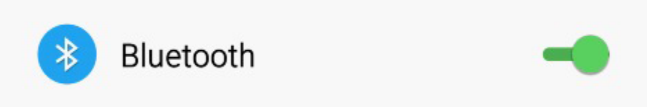
If your Phone or Tablet already has Bluetooth functionality built in then you may first have to enable it by clicking on Settings -> Connections.

Bluetooth Switched Off



Bluetooth

Bluetooth Switched On

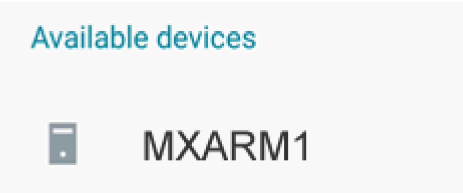


Bluetooth

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

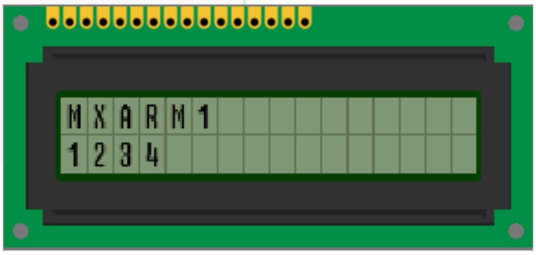
Begin by clicking the Bluetooth option in Settings -> Connections

Next make sure your robot is switched on and 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.



Available devices

MXARM1



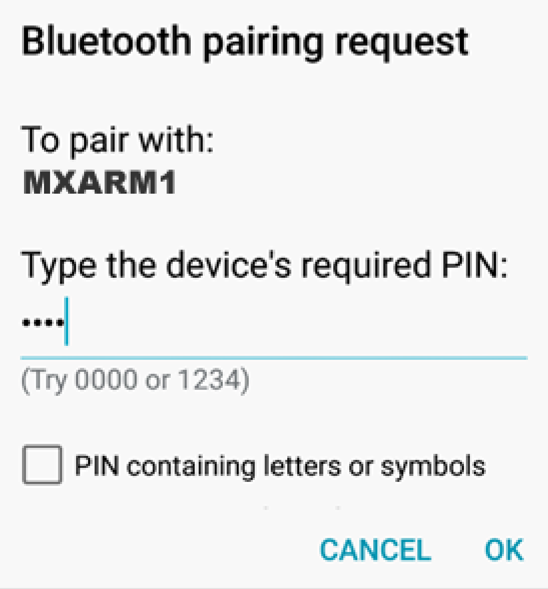
The name of the AllCode Robot Arm appears on the LCD when powering on the Robot Arm to let you know the Bluetooth name of the robot you want to connect to.

Hardware setup

Android Tablet / Phone

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.



Bluetooth pairing request

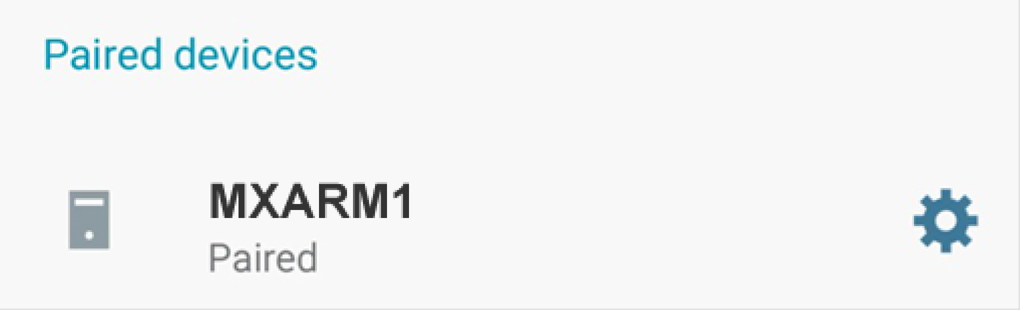
To pair with:
MXARM1

Type the device's required PIN:
••••

(Try 0000 or 1234)

□ PIN containing letters or symbols

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 AllCode Robot Arm apps you download or create.



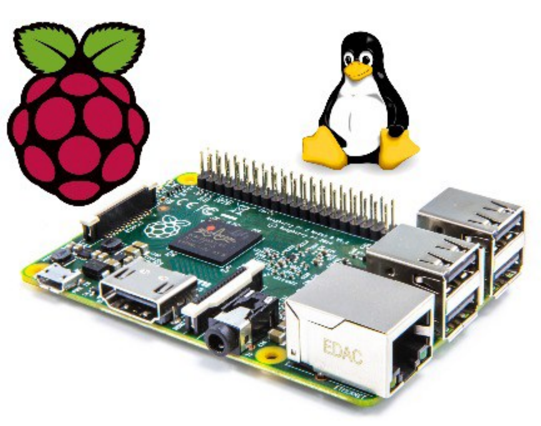
Paired devices

MXARM1
Paired

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.

Hardware setup

Raspberry Pi and Linux



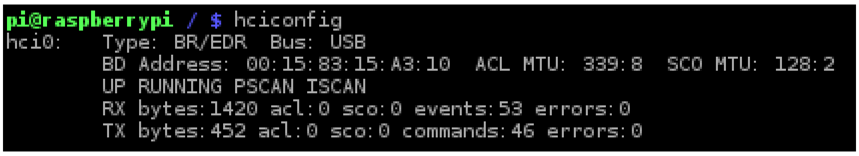
The Raspberry Pi is a popular single-board computer.

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

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

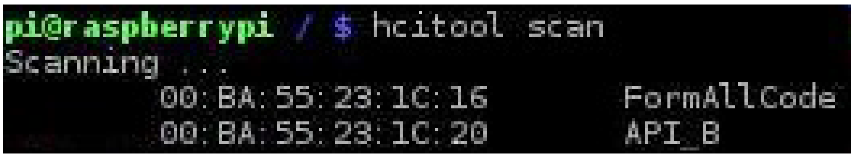
Setting up Bluetooth is relatively easy on a Raspberry Pi and can be done in a number of ways. The following steps are perhaps a more complex way of setting it up, but it should work in all situations. Note the Pi needs a Bluetooth USB dongle.

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":



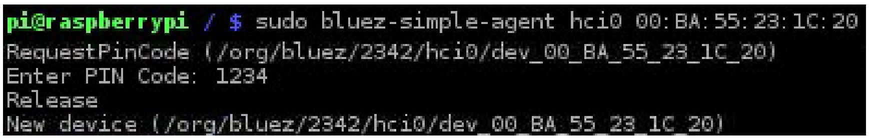
```
pi@raspberrypi / $ hciconfig
hci0:   Type: BR/EDR  Bus: USB
        BD Address: 00:15:83:15:A3:10  ACL MTU: 339:8  SCO MTU: 128:2
        UP RUNNING PSCAN ISCAN
        RX bytes:1420 acl:0 sco:0 events:53 errors:0
        TX bytes:452 acl:0 sco:0 commands:46 errors:0
```

Step 2 – Detect the AllCode Robot Arm
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,



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

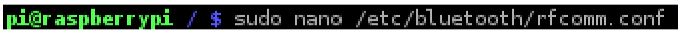
"00:BA:55:23:1C:20".



```
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)
```

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 "AllCode Robot Arm";
}
```

The three red bits of text can be customised - you will use the MAC address found in step 2, and can use the 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 command line terminal:

```
echo "LEDWrite 85\n" > /dev/rfcomm1
```

If all goes well, the LEDs on the Combo Board should light up in an alternating pattern.

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 and the "LEDWrite" command should now work ok.

Controlling the robot

Using Flowcode

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

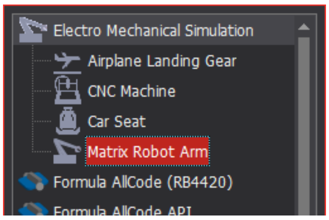
As you probably know, Flowcode provides component macros for all complex devices like CAN bus, ZigBee, and the robot. This means you can start learning about robotics and how to control the Formula AllCode Robot very quickly and very easily.

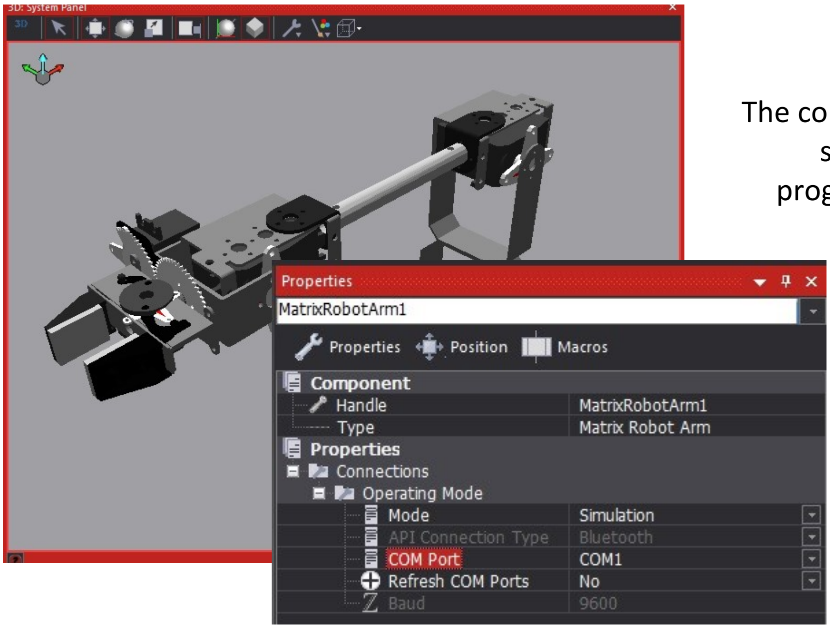
There are programs on the Matrix website to inspire and help you.

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

This section assumes you are familiar with the basics of using Flowcode. There are two ways of using Flowcode to control the AllCode Robot Arm. 1) using the in-built API functionality and 2) re-programming the firmware on the robot.

In Flowcode 7.2 and later there is a component available from the mechatronics menu to allow you to easily control the Robot Arm via the command API.





MatrixRobotArm1

Component	
Handle	MatrixRobotArm1
Type	Matrix Robot Arm
Properties	
Connections	
Operating Mode	
Mode	Simulation
API Connection Type	Bluetooth
COM Port	COM1
Refresh COM Ports	No
Baud	9600

The component comes with a fully operational simulation allowing us to create and test programs before we move to the hardware.

The component's Mode property allows us to decide if we are using the component to control the simulation or the real hardware via the API.

The component's COM port property allows us to enter the number of the COM port we collected during pairing.

Downloading code to the robot

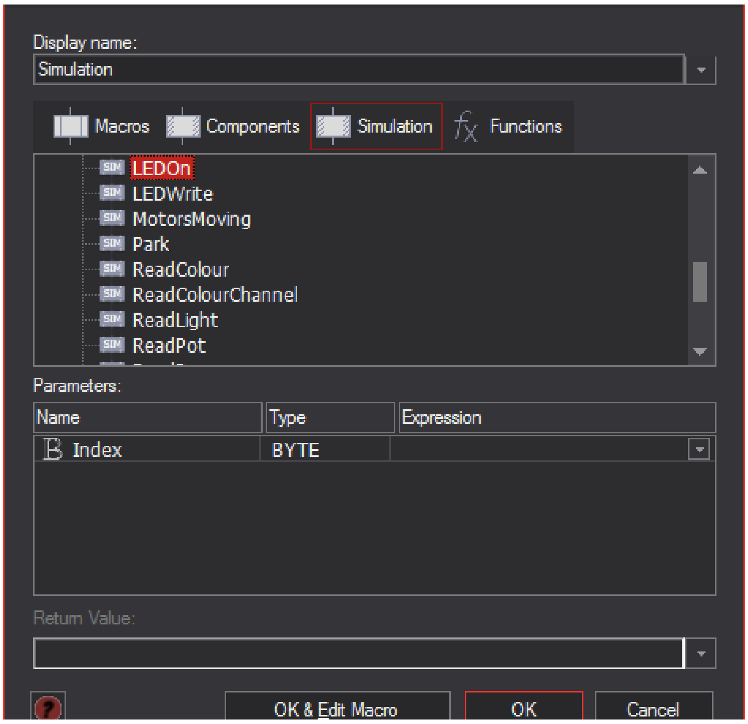
Flowcode also allows us to create code which will run on the microcontroller on-board the EB091 dsPIC E-block. This is not recommended unless you know what you're doing.

Note: Downloading code to the EB091 dsPIC E-block will remove the in-built API functionality from the robot. Instructions on restoring the API firmware to get the robot back to the original factory functionality are provided on page 4.

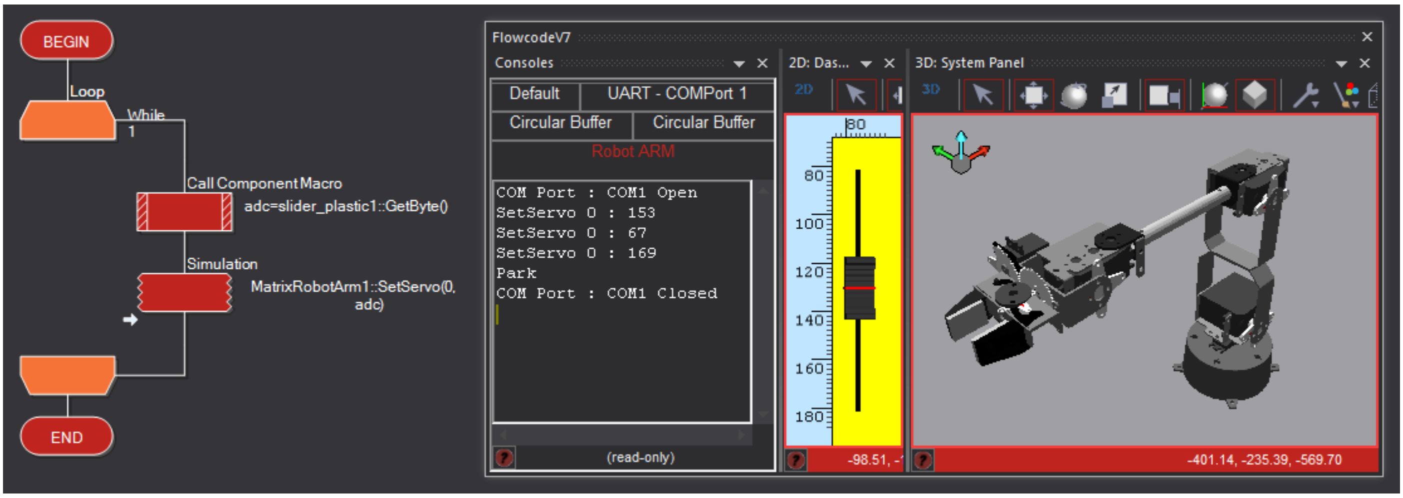
Controlling the robot

Direct control from Flowcode
The component ("Matrix Robot Arm") allows us to control an AllCode Robot Arm 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 component code is not downloadable onto the AllCode EB091 dsPIC E-block.


Display name: Simulation

Here we control the position of the Base Motor 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.



```
COM Port : COM1 Open
SetServo 0 : 153
SetServo 0 : 67
SetServo 0 : 169
Park
COM Port : COM1 Closed
```

The selected communications (COM) port is automatically opened when you start the simulation. The motors automatically return the arm to the parked position before closing the port when you end the simulation.

Controlling the robot

Using App Inventor

Robotics Course Instructional Guide

App Inventor

Template

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 QR codes and hyperlinks will help speed-up your installation, so you can start having some fun coding.

App Inventor

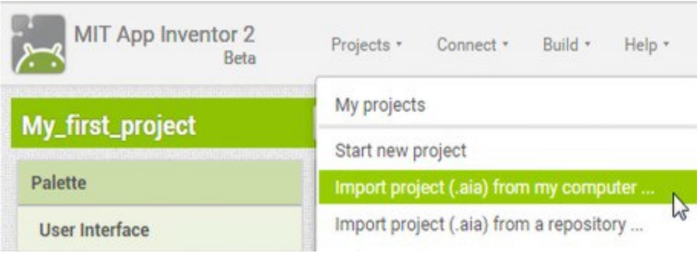
App Inventor is a freely available graphical programming language hosted on one of the cloud-computing and storage systems at Massachusetts Institute of Technology (MIT) in the United States. All you need to get started developing apps for an Android mobile phone or tablet is a web browser and a Google 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 AllCode Robot Arm template onto your computer by scanning the QR code or clicking the hyperlink. Remember where you saved them on your desktop/laptop.



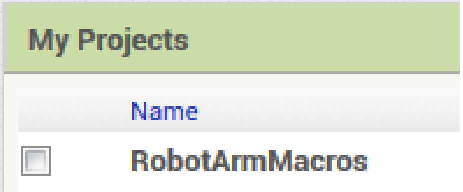
Controlling the robot

Using App Inventor

Your first program

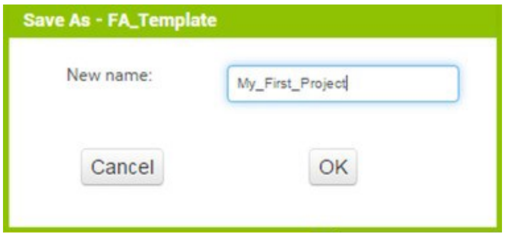
Each time you want to start a new project, follow these steps:

1. Load the template file by clicking ‘My Projects’ from the App Inventor menu and selecting the ‘RobotArmMacros’ project.



My Projects

	Name
□	RobotArmMacros



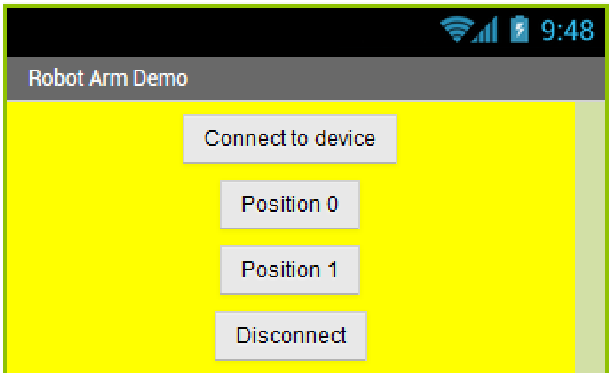
Save As - FA_Template

New name: My_First_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.

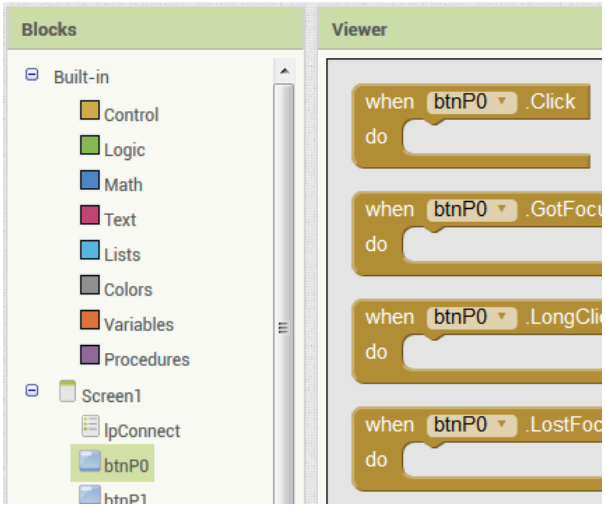
4. Drag a button from the User Interface panel onto the Viewer screen and alter its text to read “Position 0”. Also rename the button so it reads “btnP0”. Do the same again to create another button called “Position 1” with the name “btnP1”.



5. Switch to ‘Blocks’ mode and click on the ‘btnP0’ object - a list of icons will appear.

Drag the “when btnP0.Click” icon onto your program.

Do the same again but this time click on the ‘btnP1’ object.

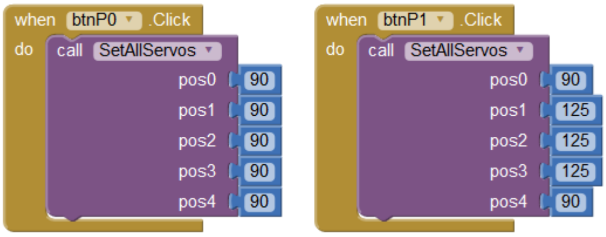


Controlling the robot

Using App Inventor

6. Click ‘Procedures’ from the ‘Built-in’ list and drag the “call SetAllServos” icon into the middle of your “when btnP0.Click” and “when btnP1.Click” icons. Add literal values from ‘Math’ as the angle parameters to the SetAllServos code blocks.

Your two blocks should look something like this:



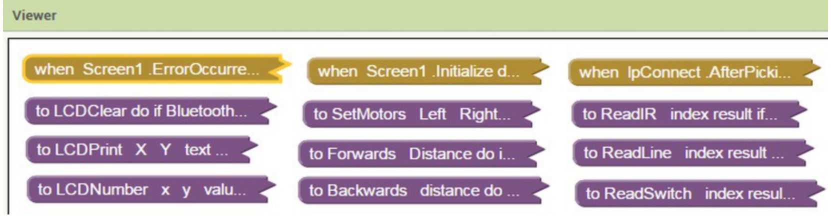
```
when btnP0 .Click
do  call SetAllServos
              pos0  90
              pos1  90
              pos2  90
              pos3  90
              pos4  90

when btnP1 .Click
do  call SetAllServos
              pos0  90
              pos1  125
              pos2  125
              pos3  125
              pos4  90
```

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 “Position 0” button should make the robot stand up with the arm out at a right angle, and “Position 1” should make the arm point straight upwards.

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 AllCode Robot Arm and some standard functions to allow the Bluetooth link to be set-up.



```
when Screen1 .ErrorOccurre...     when Screen1 .Initialize d...     when lpConnect .AfterPicki...
to LCDClear do if Bluetooth...    to SetMotors Left Right...        to ReadIR index result if...
to LCDPrint X Y text ...          to Forwards Distance do i...      to ReadLine index result ...
to LCDNumber x y valu...          to Backwards distance do ...      to ReadSwitch index resul...
```

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.

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 AllCode Robot Arm using some of the more widely known programming languages such as C++, C# and Visual Basic.

Using the AllCode Robot Arm 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 AllCode Robot Arm pages of the Matrix TSL website here: http://www.matrixtsl.com/allcode/resources/

Using C#

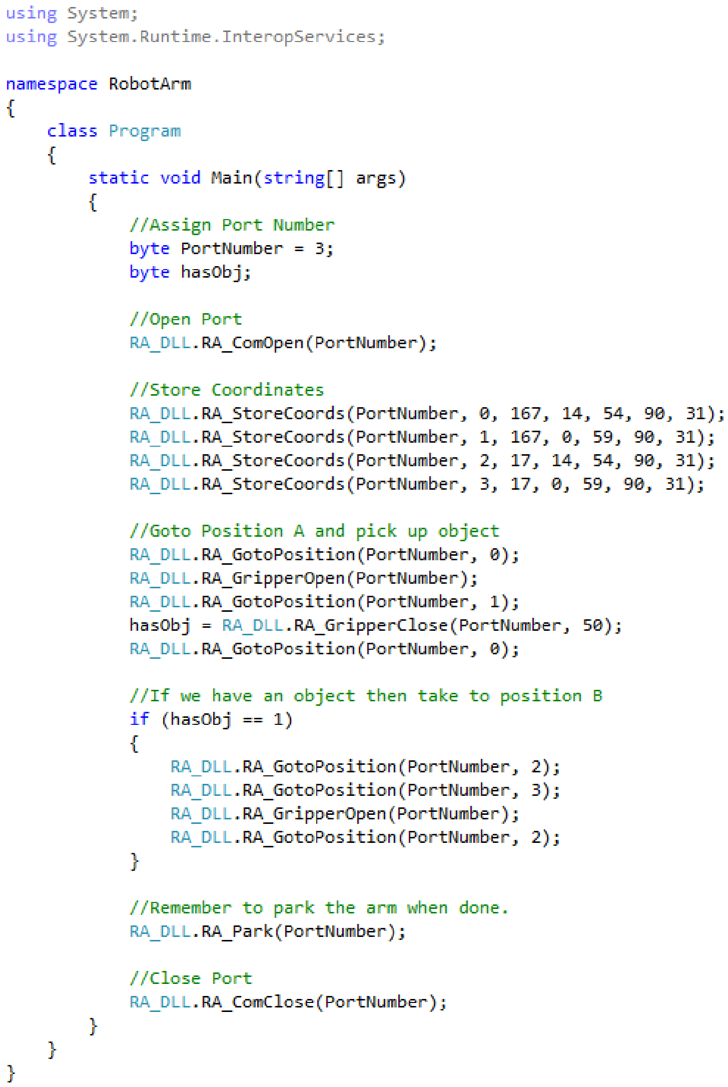
The program on the right shows a basic program in C#.

You should use the namespace “RobotArm” and place the RA_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 AllCode API commands are prefixed with the characters “RA_” 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 park the arm and close the COM port at the end of your program!



```
using System;
using System.Runtime.InteropServices;

namespace RobotArm
{
    class Program
    {
        static void Main(string[] args)
        {
            //Assign Port Number
            byte PortNumber = 3;
            byte hasObj;

            //Open Port
            RA_DLL.RA_ComOpen(PortNumber);

            //Store Coordinates
            RA_DLL.RA_StoreCoords(PortNumber, 0, 167, 14, 54, 90, 31);
            RA_DLL.RA_StoreCoords(PortNumber, 1, 167, 0, 59, 90, 31);
            RA_DLL.RA_StoreCoords(PortNumber, 2, 17, 14, 54, 90, 31);
            RA_DLL.RA_StoreCoords(PortNumber, 3, 17, 0, 59, 90, 31);

            //Goto Position A and pick up object
            RA_DLL.RA_GotoPosition(PortNumber, 0);
            RA_DLL.RA_GripperOpen(PortNumber);
            RA_DLL.RA_GotoPosition(PortNumber, 1);
            hasObj = RA_DLL.RA_GripperClose(PortNumber, 50);
            RA_DLL.RA_GotoPosition(PortNumber, 0);

            //If we have an object then take to position B
            if (hasObj == 1)
            {
                RA_DLL.RA_GotoPosition(PortNumber, 2);
                RA_DLL.RA_GotoPosition(PortNumber, 3);
                RA_DLL.RA_GripperOpen(PortNumber);
                RA_DLL.RA_GotoPosition(PortNumber, 2);
            }

            //Remember to park the arm when done.
            RA_DLL.RA_Park(PortNumber);

            //Close Port
            RA_DLL.RA_ComClose(PortNumber);
        }
    }
}
```

Controlling the robot

Using C++ / C# / VB

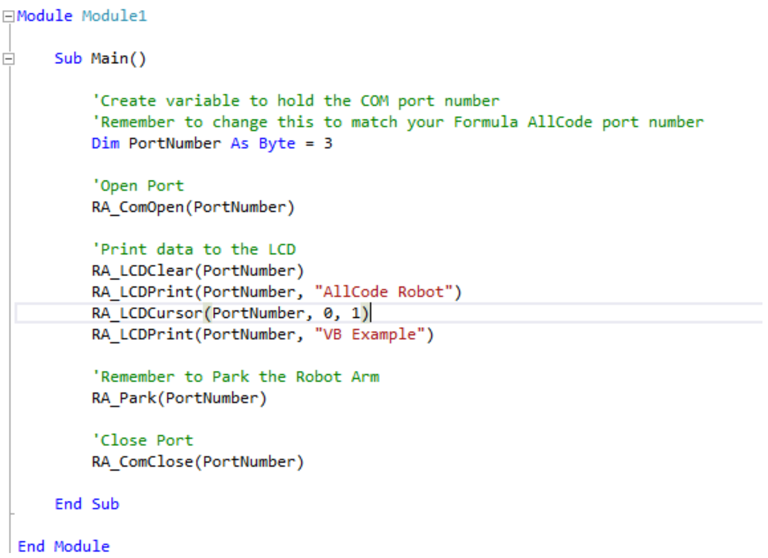
Using VB

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

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

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

Remember also to put the “RobotArm.DLL” file into the same folder as your created EXE.



```
Module Module1

    Sub Main()

        'Create variable to hold the COM port number
        'Remember to change this to match your Formula AllCode port number
        Dim PortNumber As Byte = 3

        'Open Port
        RA_ComOpen(PortNumber)

        'Print data to the LCD
        RA_LCDClear(PortNumber)
        RA_LCDPrint(PortNumber, "AllCode Robot")
        RA_LCDCursor(PortNumber, 0, 1)
        RA_LCDPrint(PortNumber, "VB Example")

        'Remember to Park the Robot Arm
        RA_Park(PortNumber)

        'Close Port
        RA_ComClose(PortNumber)

    End Sub

End Module
```

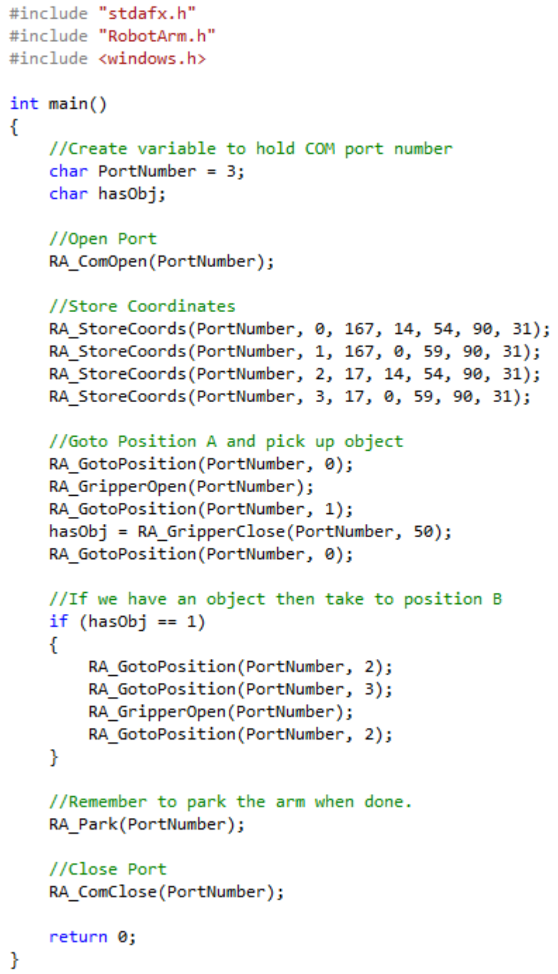
Using C++

A slightly different program is shown for C++, this time picking an item from location A and carrying it to location B.

To use the DLL with C++, you need to reference the functions by including the “RobotArmI.h” header file. You also need to add the “RobotArm.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 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 "RobotArm.h"
#include <windows.h>

int main()
{
    //Create variable to hold COM port number
    char PortNumber = 3;
    char hasObj;

    //Open Port
    RA_ComOpen(PortNumber);

    //Store Coordinates
    RA_StoreCoords(PortNumber, 0, 167, 14, 54, 90, 31);
    RA_StoreCoords(PortNumber, 1, 167, 0, 59, 90, 31);
    RA_StoreCoords(PortNumber, 2, 17, 14, 54, 90, 31);
    RA_StoreCoords(PortNumber, 3, 17, 0, 59, 90, 31);

    //Goto Position A and pick up object
    RA_GotoPosition(PortNumber, 0);
    RA_GripperOpen(PortNumber);
    RA_GotoPosition(PortNumber, 1);
    hasObj = RA_GripperClose(PortNumber, 50);
    RA_GotoPosition(PortNumber, 0);

    //If we have an object then take to position B
    if (hasObj == 1)
    {
        RA_GotoPosition(PortNumber, 2);
        RA_GotoPosition(PortNumber, 3);
        RA_GripperOpen(PortNumber);
        RA_GotoPosition(PortNumber, 2);
    }

    //Remember to park the arm when done.
    RA_Park(PortNumber);

    //Close Port
    RA_ComClose(PortNumber);

    return 0;
}
```

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 AllCode Robot Arm. 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 AllCode Robot Arm, 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 AllCode Robot Arm Python library from here: http://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

```
import RA                   # Import the Formula AllCode library
ra = RA.Create()            # Create an instance of the API
ra.ComOpen(3)               # Open the COM port
ra.LEDWrite(85)             # Assign LED pattern 0b01010101
ra.ComClose()               # Close the COM port
```

This very simple program lights up the LEDs in an alternating pattern using the API command `LEDWrite`, 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 “3” 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 RA 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 RA
import time

#Create and open 2 robots
ra1 = RA.Create()
ra2 = RA.Create()
ra1.ComOpen(3)
ra2.ComOpen(4)

#control the LEDs
ra1.LEDWrite(85)
ra2.LEDWrite(170)

# Close the COM ports
ra1.ComClose()
ra2.ComClose()
```

Going further
We have shown only a few brief examples of how to control the AllCode Robot Arm 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.

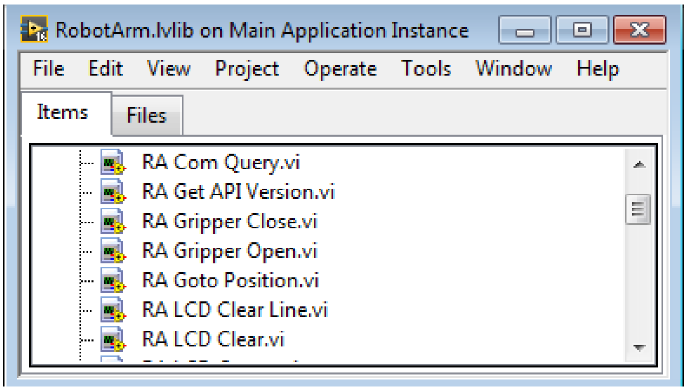
Controlling the robot

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 AllCode Robot Arm.

This section explains how to get started



RobotArm.lvlib on Main Application Instance

File Edit View Project Operate Tools Window Help

Items Files

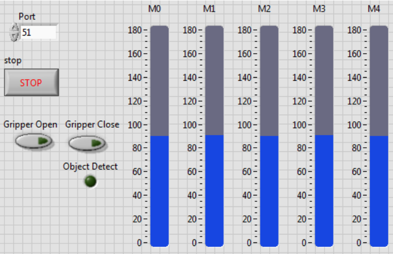
- RA Com Query.vi
- RA Get API Version.vi
- RA Gripper Close.vi
- RA Gripper Open.vi
- RA Goto Position.vi
- RA LCD Clear Line.vi
- RA LCD Clear.vi

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:

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

To begin, create a new blank VI and then open the “RobotArm.lvlib” file that was downloaded earlier. This contains all of the function calls to the 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.

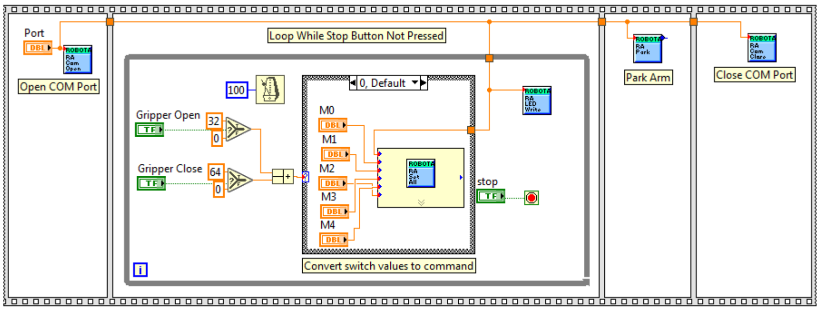


The program at the bottom of the page shows a flat sequence structure that ensures the various parts of the program are called in turn.

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 SetAllMotors API command every 100ms.

The next window places the motors into the Parked position when the loop has completed and the final right-most window closes the COM port.



Demonstration 1

Pick and Place

In this demonstration the robot arm will:

- Pick up an item from location A
- Transfer the item to location E
- Return to the parked position
- Repeat moving from E to A

A comparable real world task:

- Items arrive unbaked on a tray
- Items are picked up from the tray
- Items are placed into an oven
- The oven bakes the items
- Items are placed back onto a tray
- Tray is the moved to the next stage

To load the demo ensure the arm is in the parked position and then either press the reset button on the EB091 board or switch the robot arm off and on.

Press switch SB0 until Demo1 appears on the LCD

Press switch SB1 to start the demo.

Press switch SB7 to pick up an object from position A and deliver it to position E.

Press switch SB6 to pick up an object from position E and deliver it to position A.

Press switch SB2 to end the demo and return to the main menu.

Starting from the parked position time how long it takes to move an object between positions.

	Attempt 1	Attempt 2	Attempt 3	Attempt 4
Time Taken				

Demonstration 2

Scan and Sort

In this demonstration the robot arm will pick up an item in location A, carry the item over to the colour sensor, perform a colour scan and then transfer the item to the appropriate colour bin.

This is similar to how a real world robot arm would do a task such as picking items from a conveyor belt and sorting them based on a sensor reading.

Some example sensors would be colour, shape, bar-code etc.

Note to speed things up the industrial robot would have the sensor mounted on the end actuator. The less work there is to do the faster things can get done.

To load the demo ensure the arm is in the parked position and then either press the reset button on the EB091 board or switch the robot arm off and on.

Press switch SB0 until Demo2 appears on the LCD

Press switch SB1 to start the demo.

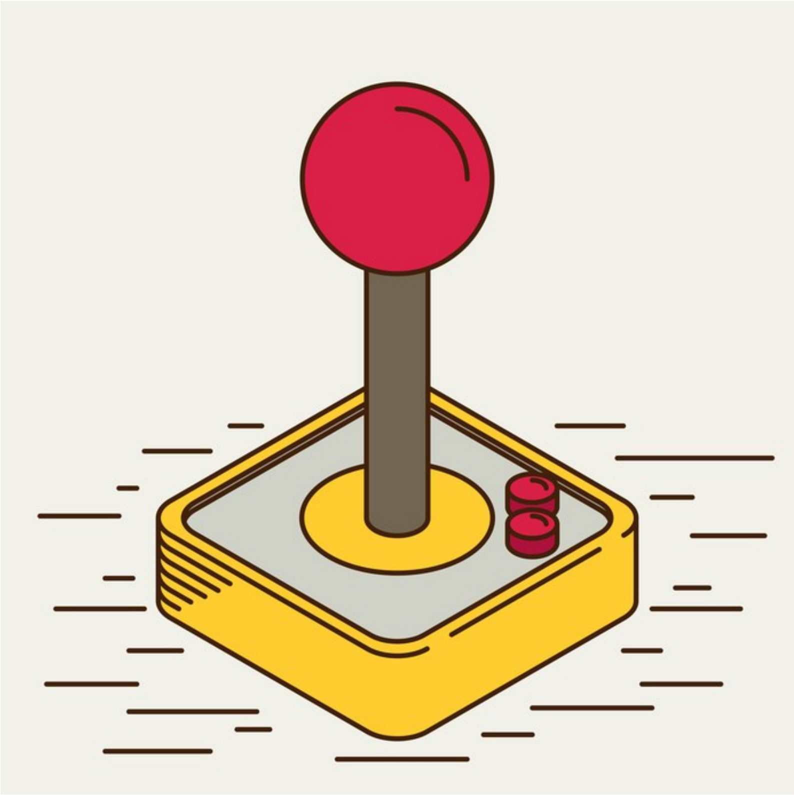
Press switch SB7 to pick up an object from position A, scan the colour and deliver it to the appropriate position.

Press switch SB6 to pick up an object from position E, scan the colour and deliver it to the appropriate position.

Press switch SB2 to end the demo and return to the main menu.

Worksheet 1

Manual Movement



Introduction:

In this exercise we will be controlling the robot directly using manual control.

- Understand the various arm joints
- Move the arm to various defined positions
- Race to perform a set operation

Setup:

To load the manual control mode ensure the arm is in the parked position and then either press the reset button on the EB091 board or switch the robot arm off and on.

Press switch SB0 until “**Manual**” appears on the LCD.

Press switch SB1 to start the manual control.

Background:

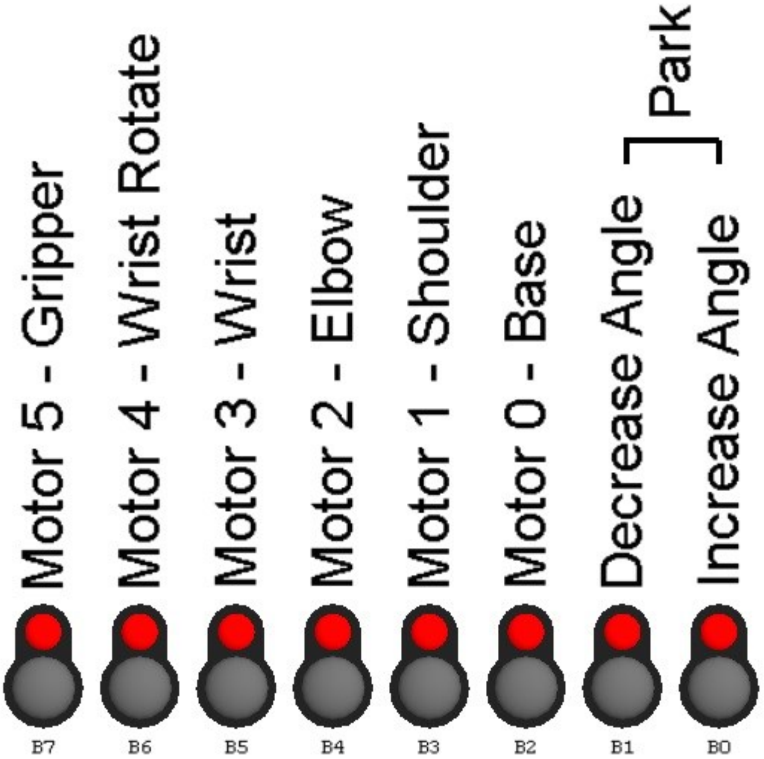
The robot arm has six motors which each control a single rotational joint. By varying the angle of the motors we can control the position of the arm.

The robot arm is controlled by the bank of eight switches in manual mode.

Switches SB2 to SB7 change which motor is currently being controlled. The LCD and PORT A LEDs show you the motor that is currently selected.

Switches SB0 and SB1 control the position of the selected motor.

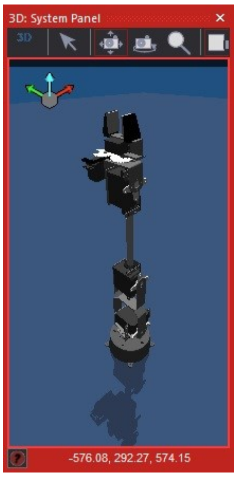
Pressing SB0 and SB1 at the same time automatically moves the robot arm to the park position and returns to the main menu.



Worksheet 1

Manual Movement

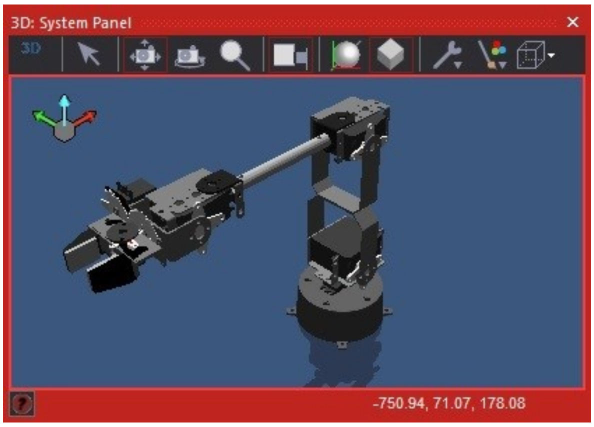
Robotic Arm Course Instructional Guide



-576.08, 292.27, 574.15

A - Try and move the arm so it is pointing straight up

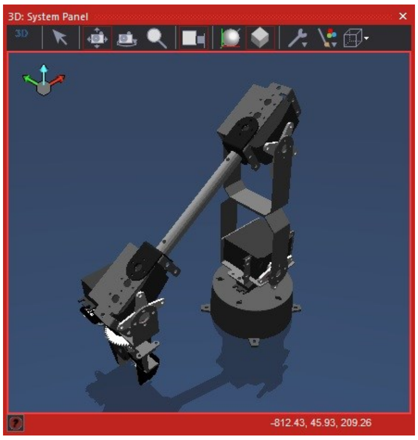
Record your angles in the table below.



-750.94, 71.07, 178.08

B - Try and move the arm so it is reaching out at 90°.

Record your angles in the table below.



-812.43, 45.93, 209.26

C - Try and move the arm so that the gripper is touching the mat and pointing vertically downwards

	Angle 0	Angle 1	Angle 2	Angle 3	Angle 4
Straight Up					
90 Degrees					
Gripper Down					

Further Work - Starting from the parked position time how long it takes to move to all three positions and then return to the parked position. Your recorded angles should help to speed things up.

	Attempt 1	Attempt 2	Attempt 3	Attempt 4
Time Taken				

Worksheet 2

Recording Positions

Introduction:

In this exercise we will be controlling the robot directly using manual control.

- Collect the angles for defined positions
- Work out the force required to pick up and hold an object
- Race to perform a set operation

Setup:

To load the manual control mode ensure the arm is in the parked position and then either press the reset button on the EB091 board or switch the robot arm off and on.

Press switch SB0 until "**Manual**" appears on the LCD.

Press switch SB1 to start the manual control.

Background:

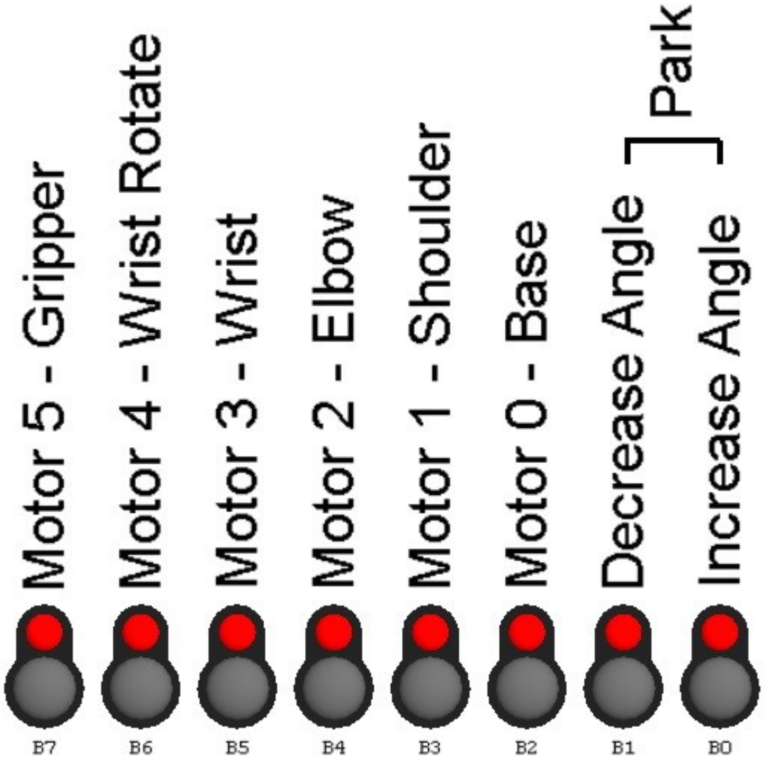
The robot arm mat has a number of marked locations to denote pick-up and drop-off points. By recording the arm's motor angles when the gripper is in position over the drop-off points we will be able to return to the positions again and again.

The robot arm is controlled by the bank of eight switches in manual mode.

Switches SB2 to SB7 change which motor is currently being controlled. The LCD and top row LEDs show you the selected motor.

Switches SB0 and SB1 control the position of the selected motor.

Pressing SB0 and SB1 at the same time automatically moves the robot arm to the park position and returns to the main menu.



Worksheet 2

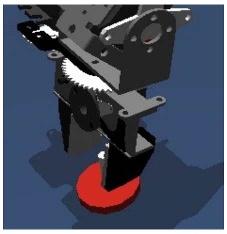
Recording Positions

A - Place one of the coloured marker objects onto position A and another onto position B.

B - Open the gripper so that it is fully open.

C - Move the arm until the gripper is above the object on position A.

Record all of the motor angles in the table below.



D - Move the arm until the gripper is positioned on the object on position A.

Record all of the motor angles in the table below.

E - Repeat the process for Position B and record the angles.

	Index	Angle 0	Angle 1	Angle 2	Angle 3	Angle 4
Over Position A	0					
On Position A	1					
Over Position B	2					
On Position B	3					

Background:

The reason we record the angles above the target position is so that we don't disturb the object on the position when moving into position.

By first moving above the object we ensure that no matter where the arm was before we are on a good course to pick up the object without accidentally moving it out the way.

Press switches SB0 and SB1 together to park the arm.

Further Work - Time how long it takes to manually move to position A, pick up an object and move to position B before releasing the object and returning to the parked position.

	Attempt 1	Attempt 2	Attempt 3	Attempt 4
Time Taken				

Worksheet 3

API Control

Robotic Arm Course Instructional Guide

Introduction:

In this exercise we will be controlling the robot indirectly using API control.

- Establish communications
- Light an LED
- Read a Sensor
- Control a Display

Setup:

To load the API control mode ensure the arm is in the parked position and then either press the reset button on the EB091 board or switch the robot arm off and on.

The API mode is automatically selected when you start communicating with the arm.

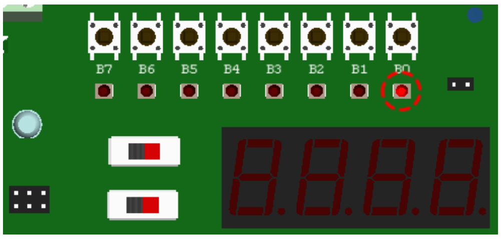
Background:

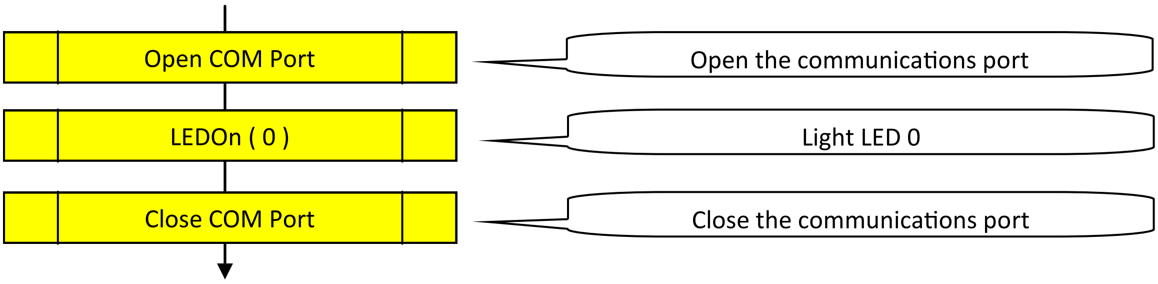
For these next exercises we will be using pseudo code to allow you to write your control program in which ever software you feel most comfortable.

Some examples include Flowcode, Python, Labview, C#, VB, Java, AppInventor etc.

A - Let’s begin by lighting a single LED

Using the pseudo code below as a guide create a program that opens the communication port, calls the LEDOn function with parameter 0 and closes the communication port.





The Open COM Port and Close COM Port commands will not be shown in the proceeding worksheet examples but are still required.

Worksheet 3

API Control

Now we can control the arm remotely let’s try something a bit more adventurous.

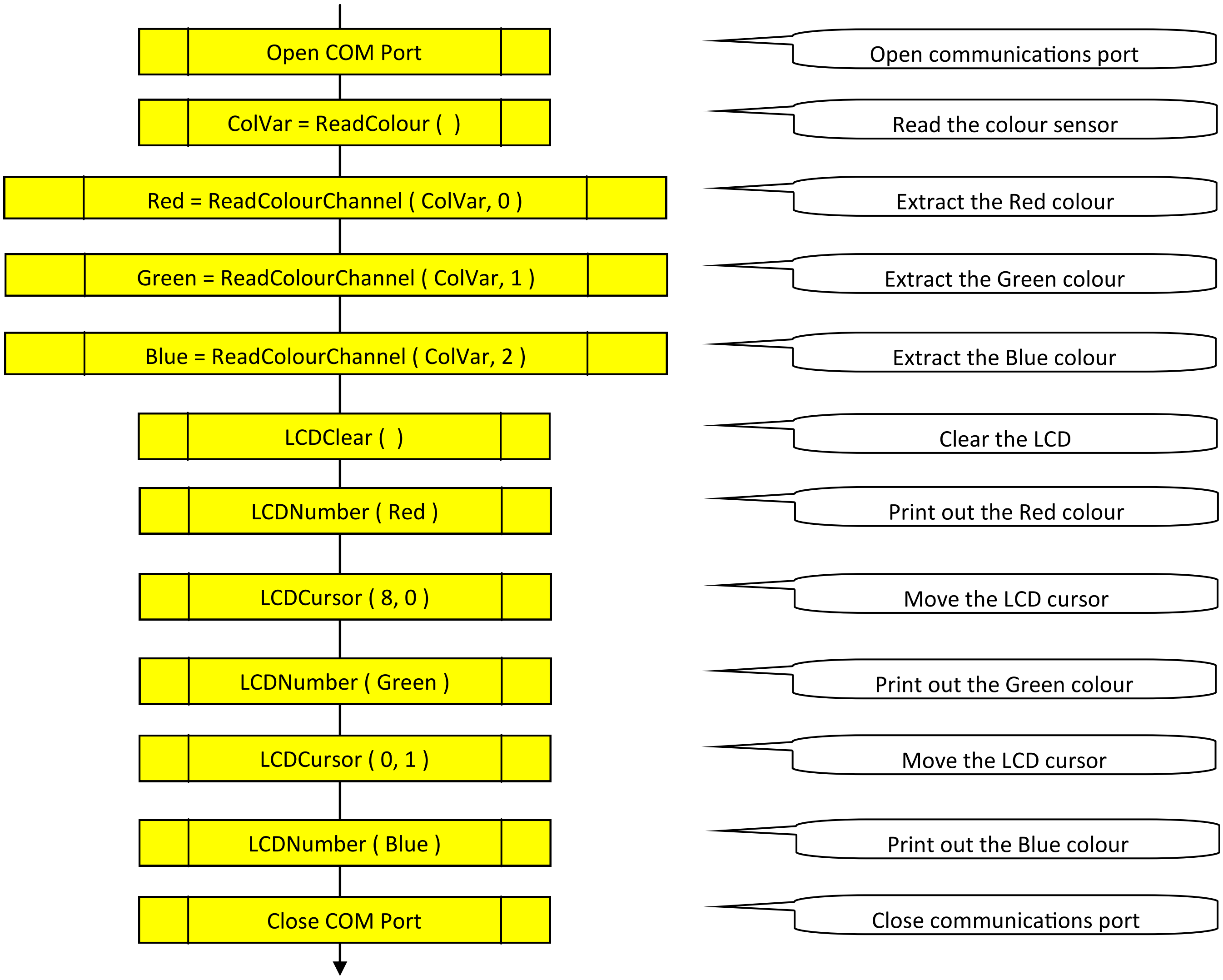
B - Read the colour sensor and print the individual Red, Green and Blue colour channels to the LCD.

Record the colour channels for the markers in the table below.

The colour is read as a 16-bit integer value using the ReadColour API command.

The individual Red, Green and Blue colour channels are obtained from the 16-bit value using the ReadColourChannel API command.

Here is the pseudocode program to read the sensor and display the readings.

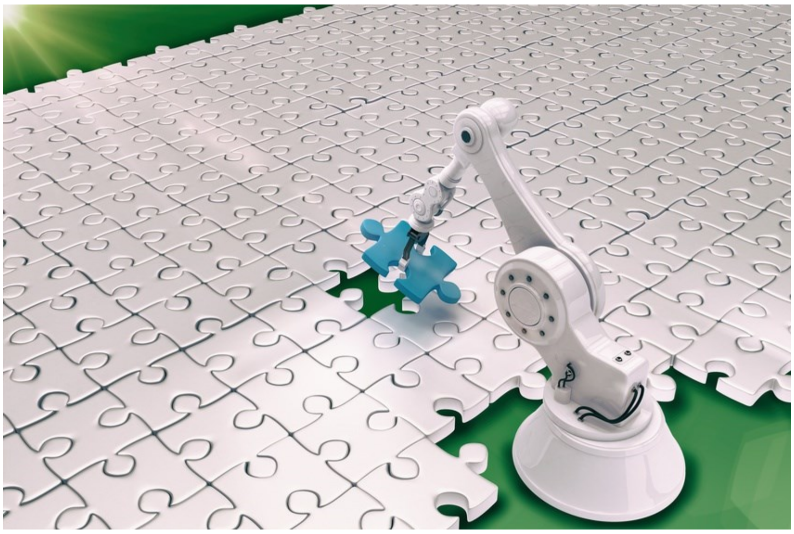


Marker	Red Channel	Green Channel	Blue Channel
Red			
Green			
Blue			

Further Work - Based on the colour channel values can you print the name of the coloured marker on the LCD.

Worksheet 4

Automatic Pick and Place



Introduction:

In this exercise we will be controlling the robot indirectly using API control.

- Establish communications
- Basic movements

Setup:

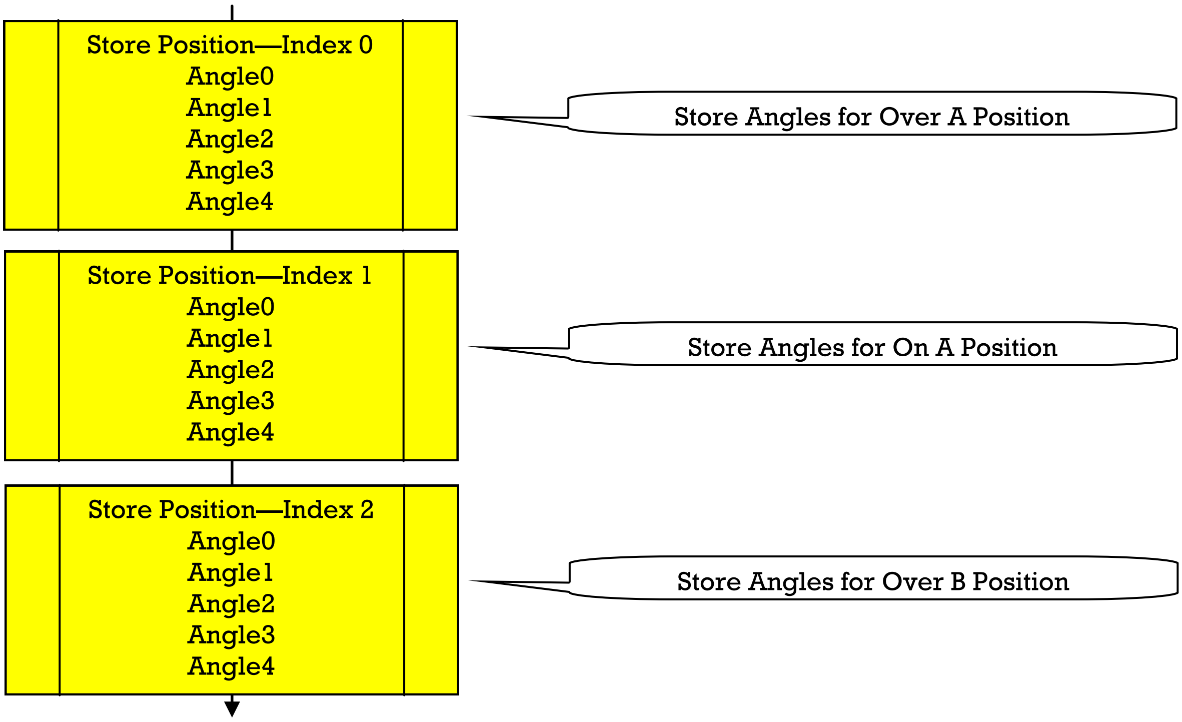
To load the API control mode ensure the arm is in the parked position and then either press the reset button on the EB091 board or switch the robot arm off and on.

Background:

Being able to reliably pick up an item and place it down is a common task for a robot arm. Some robot arms may spend their entire service life picking up items and placing them elsewhere. A pick and place machine is a machine that takes electronic components and places them in known pre-set locations on a circuit board.

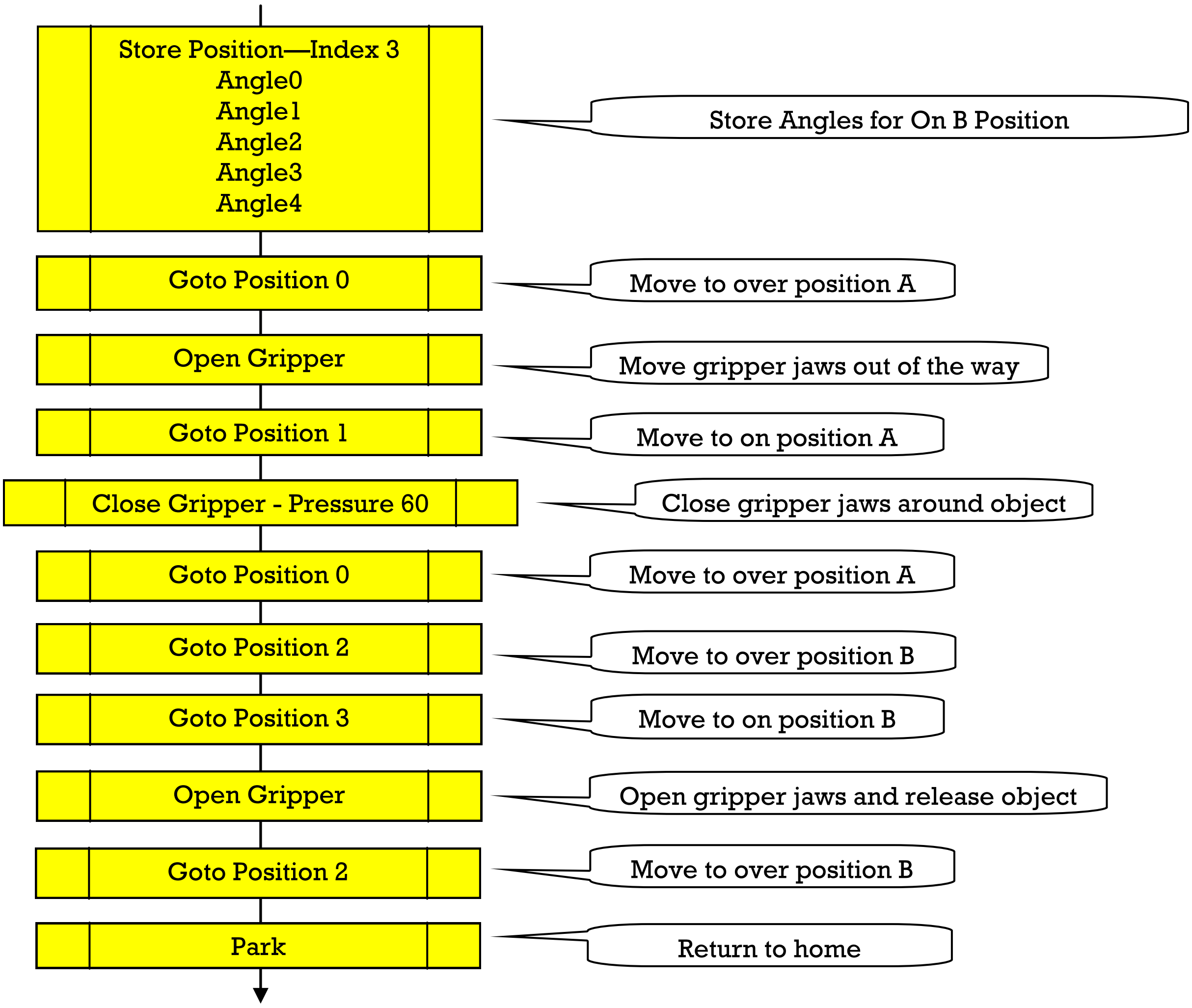
A - Lets begin by trying to pick something up and place it back down in another location. The marked locations on the Robot Arm mat allow the coloured markers to be positioned in known locations. Let’s begin by moving a marker from locations A to B.

Using the pseudo code below as a guide create a program that opens the communication port, stores the coordinates we collected earlier in Worksheet 2 and moves between the coordinates.

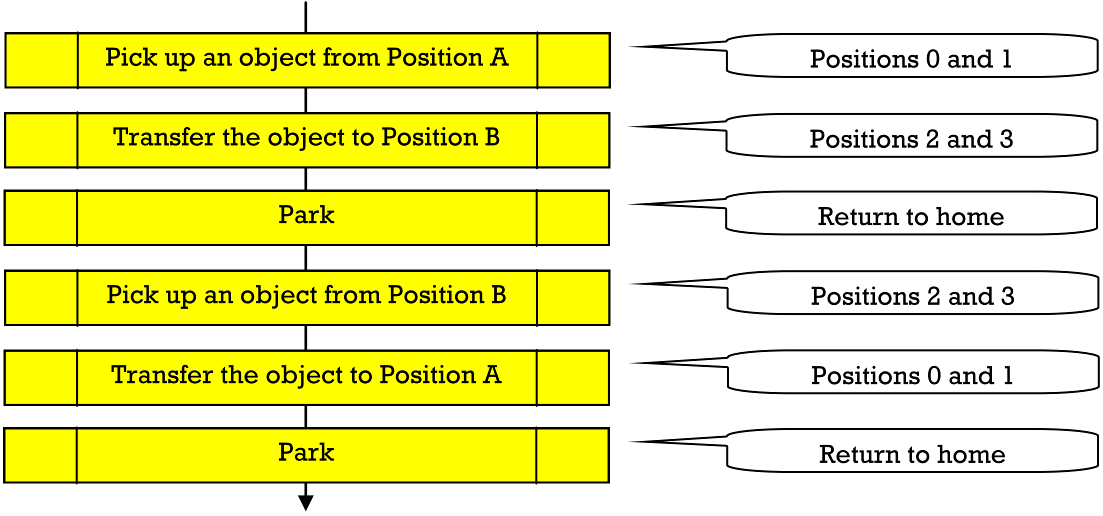


Worksheet 4

Automatic Pick and Place



B - Now we can move a marker from position A to position B lets do the same thing in reverse and move the marker back from position B to position A.

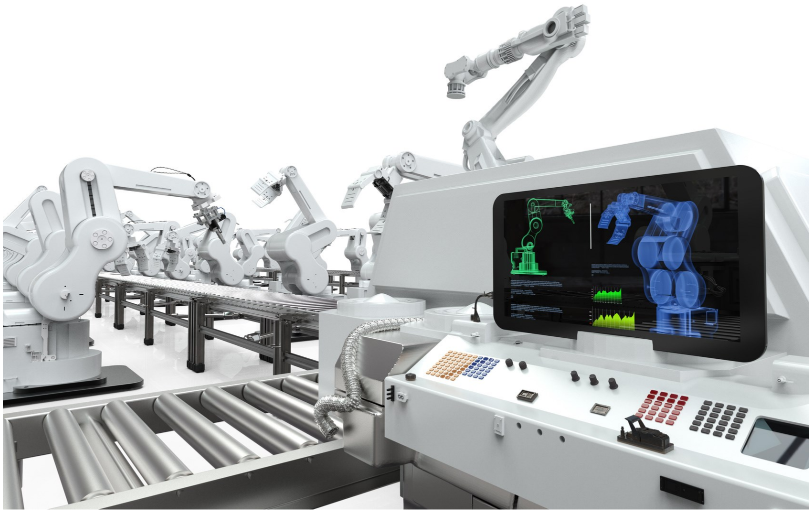

Pick up an object from Position A
Positions 0 and 1
Transfer the object to Position B
Positions 2 and 3
Park
Return to home
Pick up an object from Position B
Positions 2 and 3
Transfer the object to Position A
Positions 0 and 1
Park
Return to home

Further Work - A - Can you tell if the arm has picked up the object when closing the gripper? If no object is detected return to the parked position. **B** - Experiment with the pressure variable when closing the gripper, what is the minimum force to reliably pick up the marker and keep it stable in the gripper. **C** - Can you modify your program to repeatedly move a marker from A to B and back again. End the program when one of the SB switches are pressed.

Worksheet 5

Complex Movements

Robotic Arm Course Instructional Guide



Introduction:

In this exercise we will be controlling the robot indirectly using API control.

- Path Planning
- Controlled Movement

Setup:

To load the API control mode ensure the arm is in the parked position and then either press the reset button on the EB091 board or switch the robot arm off and on.

The API mode is automatically selected when you start communicating with the arm.

Background:

Being able to make the robot arm perform complex movements is a common task. Often there are lots of potential things which are in the way which the arm must navigate around when performing it's day to day tasks. These could be physical objects to avoid or could be more subtle such as unexpected constraints in the arms movement mechanisms.

A - Using the manual mode collect the coordinates for on and over locations C, D and E. Record the angles in the table below.

	Index	Angle 0	Angle 1	Angle 2	Angle 3	Angle 4
Over C	4					
On C	5					
Over D	6					
On D	7					
Over E	8					
On E	9					

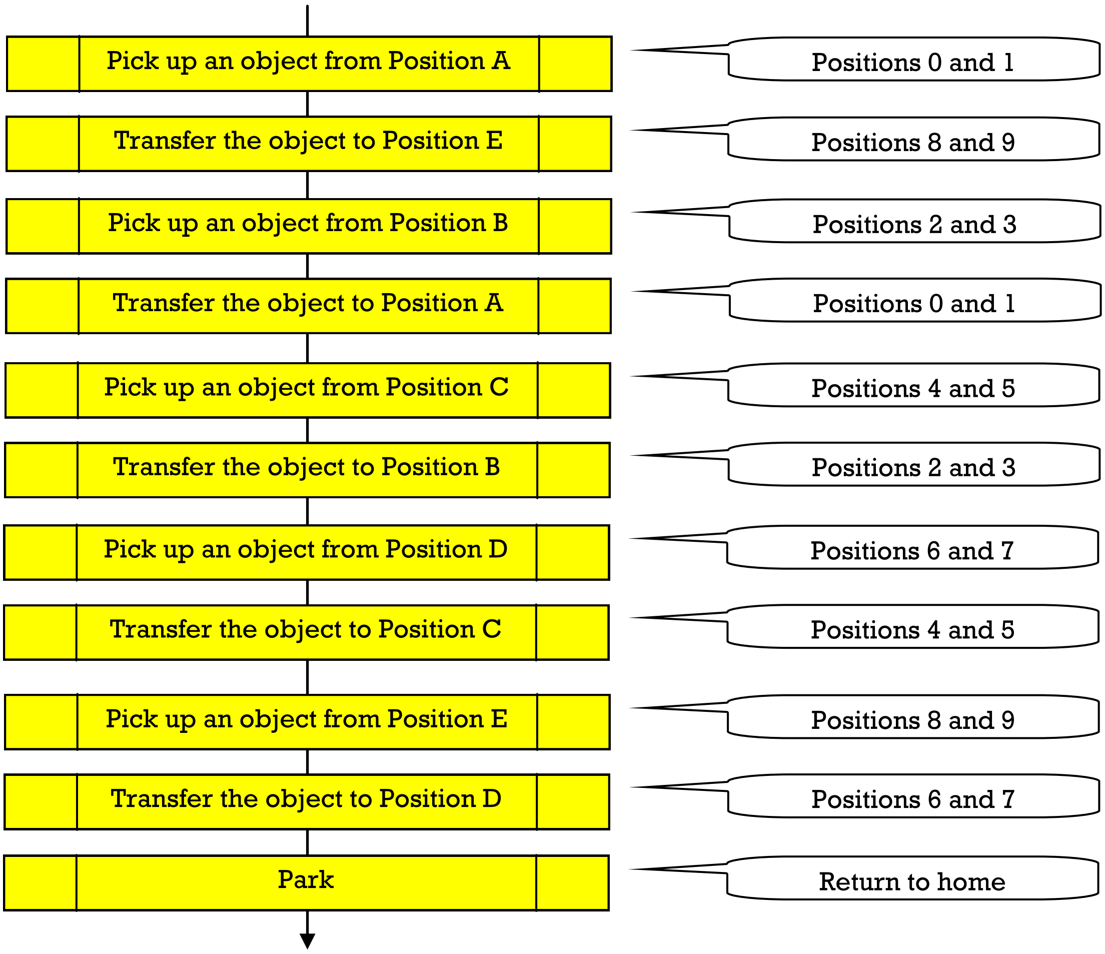
Worksheet 5

Complex Movements

Robotic Arm Course Instructional Guide

B - Position coloured markers on locations A, B, C and D leaving location E empty.

Using the API mode and following the pseudo code below see if you can move the markers round in a circle.



Pick up an object from Position A
Positions 0 and 1

Transfer the object to Position E
Positions 8 and 9

Pick up an object from Position B
Positions 2 and 3

Transfer the object to Position A
Positions 0 and 1

Pick up an object from Position C
Positions 4 and 5

Transfer the object to Position B
Positions 2 and 3

Pick up an object from Position D
Positions 6 and 7

Transfer the object to Position C
Positions 4 and 5

Pick up an object from Position E
Positions 8 and 9

Transfer the object to Position D
Positions 6 and 7

Park
Return to home

C - Try and repeat the routines so that the coloured markers travel right round the positions on the mat and end up back in their original positions.

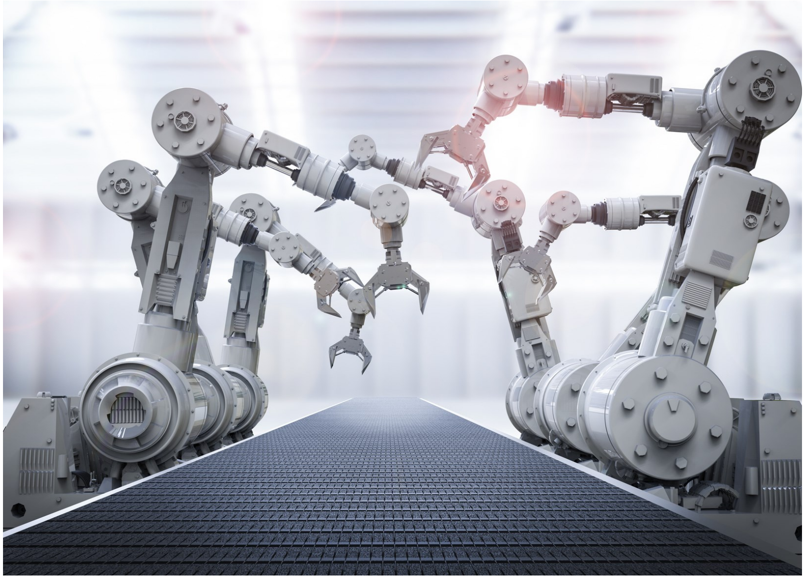
Time how long it takes to do a full cycle and then see if you can improve the time by reducing the movements. Remember that if you hit or move one of the coloured markers when moving the arm then that counts as a fail.

	Attempt 1	Attempt 2	Attempt 3	Attempt 4
Time Taken				

Further Work - **A** - Add a fifth coloured marker to the E position and see if you can move all 5 markers between positions in a circular pattern. You will likely need to use a free area on the mat as a temporary storage location.
B - Try a different movement pattern. For example move A to E, D to A, B to D, C to B and finally E back to C.

Worksheet 6

Automatic Sorting



Introduction:

In this exercise we will be controlling the robot indirectly using API control.

- Sensor Interrogation
- Decision Making

Setup:

To load the API control mode ensure the arm is in the parked position and then either press the reset button on the EB091 board or switch the robot arm off and on.

The API mode is automatically selected when you start communicating with the arm.

Background:

Being able to scan and sort objects based on a number of parameters is a very useful feature of a system.

For example in an automated bakery you might check when individual items need turning over and when they need removing from a grill.

A - Using the manual mode pick up a coloured marker from location A on the mat.

B - With the coloured marker being held transfer the marker to the colour sensor.

Record the motor angles for over and on the colour sensor.

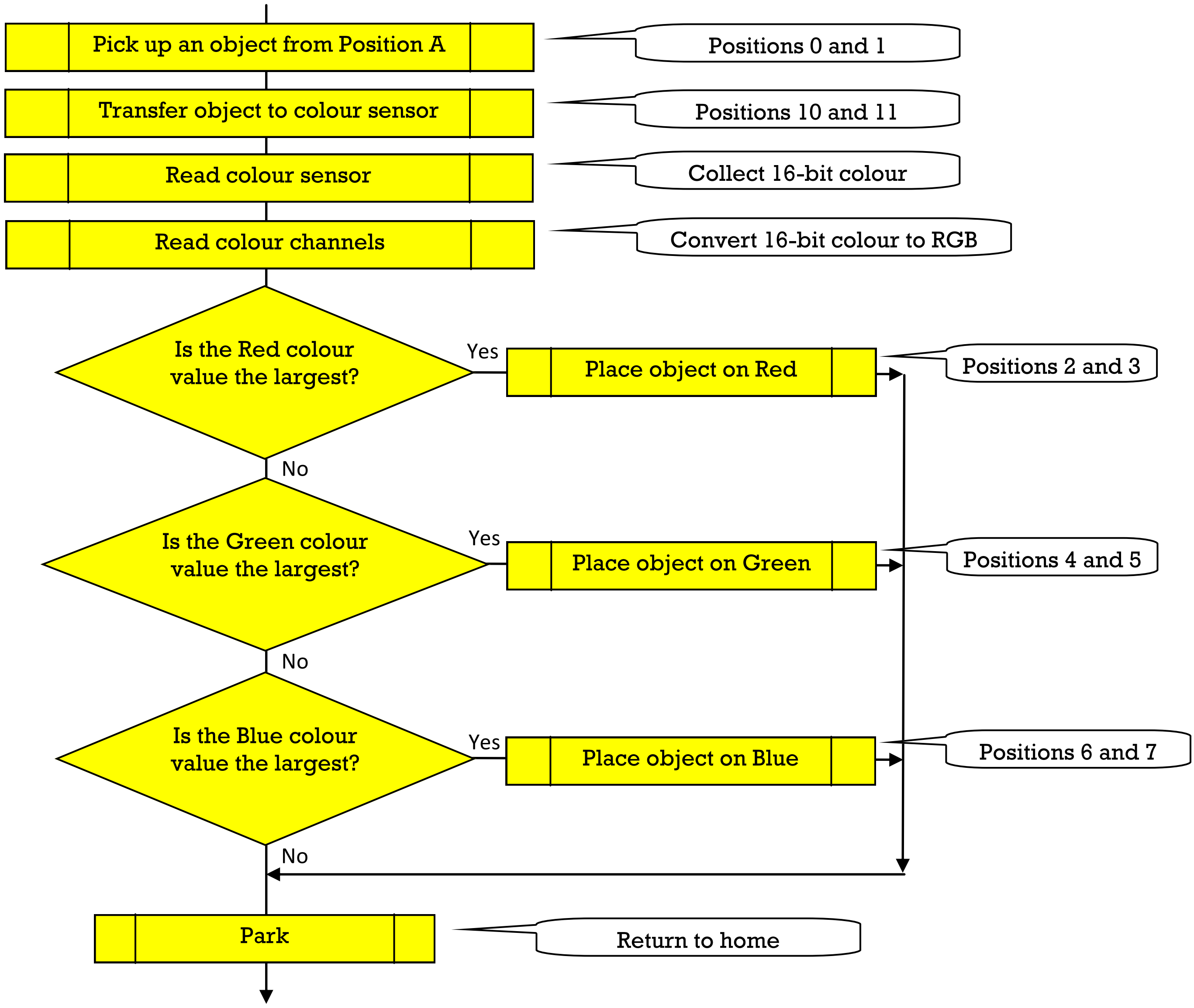
	Index	Angle 0	Angle 1	Angle 2	Angle 3	Angle 4
Over Colour Sensor	10					
On Colour Sensor	11					

Worksheet 6

Automatic Sorting

C - Using the new position angles create a program to take a coloured marker from location A on the mat, take the marker to the colour sensor, scan the colour of the marker, make a decision on where to place the marker, place the marker on the correct colour location on the mat.

Use the pseudo code below to help design your program.



Further Work - **A** - Try printing the colour of the scanned object to the LCD as a string Red, Green or Blue. **B** - Try scanning the colour of other objects. Yellow is made up of Red and Green light so how do you differentiate a Yellow object from a Red or a Green object.

Worksheet 7

Path Following

Introduction:

In this exercise we will be controlling the robot indirectly using API control.

- Path Planning
- Controlled Movement

Setup:

To load the API control mode ensure the arm is in the parked position and then either press the reset button on the EB091 board or switch the robot arm off and on.

The API mode is automatically selected when you start communicating with the arm.

Background:

Being able to make the end tool follow a given path is a common task for a robot arm. Some robot arms may spend their entire service life following strict control paths. For example a robot arm on a production line might cut, grind or weld a specific feature into a material or may be used to paint, buff or polish a high cost item such as a car. When your dealing with an expensive item the movements must be highly controlled and repeatable to avoid damaging to item.

A - Place the robot arm on a large sheet of blank paper.

Using the manual mode have the gripper hold a pen or pencil. A thick marker pen such as a white-board marker works well.

Try to move the arm to draw an single line arc on the paper.

Record the motor angles for key parts of the drawing i.e. the start and end of the line.

B - Once you have manually drawn an arc try and repeat the process using the API mode.

Use the Store Position function to save the key motor angles from part **A**.

Use the Goto Position function to move the Arm to the stored position.

The arc can easily be turned into a smiley face with the addition of two more coordinates to create the eyes.

Worksheet 7

Path Following

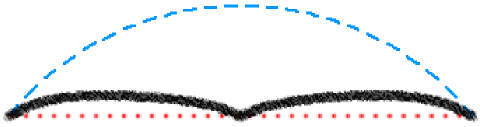
C - Next we will try to straighten the arc into a straight line.

Using a ruler draw a line from your current pen start and end points.

Using the manual mode collect set of motor angles to position the pen half way along the newly drawn line.

Using the API mode draw a line from your original point to the original end point via the new point

You should end up with something like this; with the original arc shown in blue and the ruler drawn line in red.



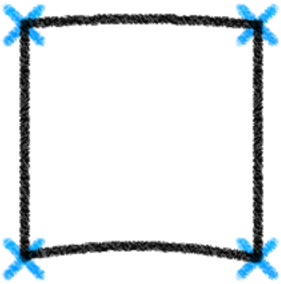
D - Now we can draw a fairly straight line while rotating the base motor how about more complex objects such as a square.

First map out the coordinates for the corners of the square on to a piece of paper.

Using the manual mode position the pen so that it is on each of the corners and record the motor angles.

If you want to take half way samples then this will improve the look of your square.

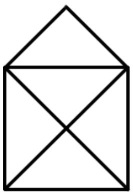
You should end up with something like this; with the corner coordinates shown in blue.



E - Try and draw some more basic shapes such as Triangle, Pentagon and Hexagon.

How do these come out and how can you improve their shape.

As a final challenge - Can you get the arm to draw this image using the API mode, without taking the pen off the paper and without going back over any lines.



Further Work - **A** - Try to use the API mode to draw something more complex. For example a circle or your name. **B** - Try to create a dock for the pen or pencil so you can automatically pick it up and place it back down. A small cardboard tube such as a tube of smarties should work well. **C** - Can you find a solution to allow the pen to remain in better contact with the paper.

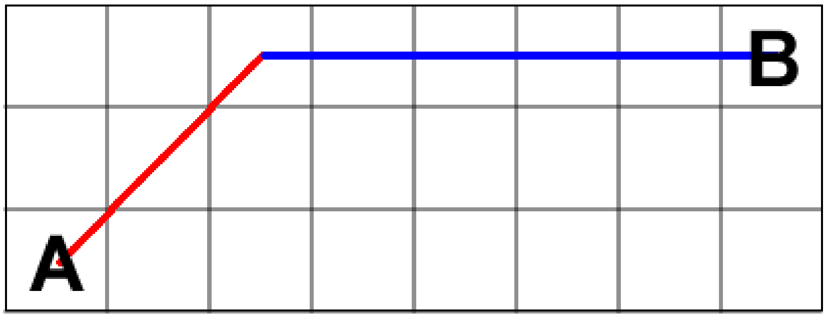
Path Planning

Linear Interpolation

Some robot arms may not have a gripper, they might have a milling cutting tool. These tools can be used to precisely cut an object. Another example would be a arm with a pen which can be used as a plotter.

For the plotter to work correctly the arm must go from it's current position to the new position using a technique called linear interpolation.

Say the arm has to travel from position A to position B. This involves a movement of 70° on one motor but only 30° on another motor. If both motors move at the same rate then the motor with the smallest angle to traverse will reach it's destination well before the other motor.

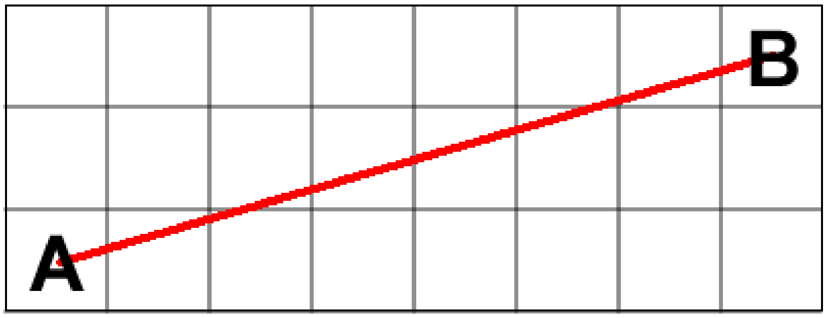


Travelling from A to B without linear interpolation

Base Motor traveling 70°
Shoulder Motor traveling 30°

Red = Two motors moving
Blue = One motor moving

By monitoring the angle differential that is required to move we can vary the individual motor speeds and ensure that they all reach their final angular position at the same time.



Travelling from A to B with linear interpolation

Base Motor traveling 70°
Shoulder Motor traveling 30°

Red = Two motors moving

Example linear interpolation maths

```
AngleDistance0 = 70                                //Assign the distances the motors need to travel to reach new angle
AngleDistance1 = 30

Max = Max( AngleDistance0, AngleDistance1 )        //Find maximum angular distance

Speed0 = AngleDistance0 / Max  = 1                 //Divide each distance by the max
Speed1 = AngleDistance1 / Max  = 0.428
```

API Documentation

Communications, LEDs and Sensors

Connection

Return	Command	Parameter(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)
	SetBTName	Name	Sets the Bluetooth device name Name = <string>
	SetBTKey	Key	Sets the Bluetooth pair key Key = <four digit numeric string>
Version	GetAPIVersion		Returns the version number of the API Version = 1 to 65535

LEDs

Return	Command	Parameter(s)	Description
	LEDWrite	Value	Writes a value to all 8 LEDs Value = 0 to 255
	LEDOn	Index	Switches on a single LED Index = 0 to 7
	LEDOff	Index	Switches off a single LED Index = 0 to 7

Sensors

Return	Command	Parameter(s)	Description

Value	ReadSwitch	Index	Reads the value of one of the eight switches
			Index = 0 to 7
			Value = 0 (Not Pressed) or 1 (Pressed)

Value	ReadLight		Reads the light sensor on the sensors board Value = 0 to 255
Value	ReadPot		Returns the potentiometer on the sensors board Value = 0 to 255
Value	ReadPressure		Returns the current pressure on the gripper jaws Value = 0 to 255
Value	ReadMagnet		Returns the output of the hall effect sensor. Value = 0 to 1
Value	ReadColour		Returns the output of the colour sensor. Value = 0 to 65535 — 0b0BBBBBGGGGGRRRRR

Value	ReadColourChannel	Index
		Colour
			Returns a single colour channel from a colour read.
			Index = 0 (Red), 1 (Green) or 2 (Blue)
			Colour = 16-bit Colour value
			Value = 0 to 255

Done	Beep	Iterations On Delay Off Delay	Controls the Buzzer on the Robot Arm Sensor Module Iterations = 1 (Once) to 255 (Max) On Delay = 1 to 65535 milliseconds Off Delay = 1 to 65535 milliseconds

API Documentation

Motors and Display

Motors

Return	Command	Parameter(s)	Description

Done	SetServo	Index
		Position
			Sets the position of a single servo motor
			Index = 0 to 4
			Position = 0 to 180
			Done = 0 (Movement Complete)

Done	SetAllServos	Position0 Position1 Position2 Position3 Position4	Sets the position of all servo motors Position(0-4) = 0 to 180 Done = 0 (Movement Complete)

Value	GripperClose	Pressure	Closes the gripper to a defined pressure
			Pressure = 1 (Weak) to 100 (Strong)
			Value = 0 (Nothing) to 1 (Something)

Done	GripperOpen		Opens the gripper Done = 0 (Movement Complete)
Done	Park		Parks the arm and switches off the motors Done = 0 (Movement Complete)
	StoreCoords	Index Position0 Position1 Position2 Position3 Position4	Stores the defined servo angles so we can easily move to the position. Index = 0 to 29 Position(0-4) = 0 to 180
	StorePosition	Index	Stores the current servo positions so we can easily move back to the position. Index = 0 to 29

Done	GotoPosition	Index	Moves the arm to a stored position.
			Index = 0 to 29
			Done = 0 (Movement Complete)

Display

Return	Command	Parameter(s)	Description
	LCDClear		Clears all data on the LCD screen.
	LCDPrint	Text	Print text on the LCD. Text = <string>
	LCDNumber	Number	Prints a number on the LCD. Number = -32768 to 32787

	LCDCursor	X
		Y
			Moves the LCD cursor to the selected X,Y position.
			X = 0 to 15
			Y = 0 to 1

	LCDClearLine	Line	Clears a single line on the LCD and returns the cursor position to the start of the line. Line = 0 to 1

Matrix Technology Solutions Ltd.
33 Gibbet Street
Halifax
HX1 5BA

t: +44 （0） 1422 252380
e: sales@matrixtsl.com

www.matrixtsl.com

CP8656