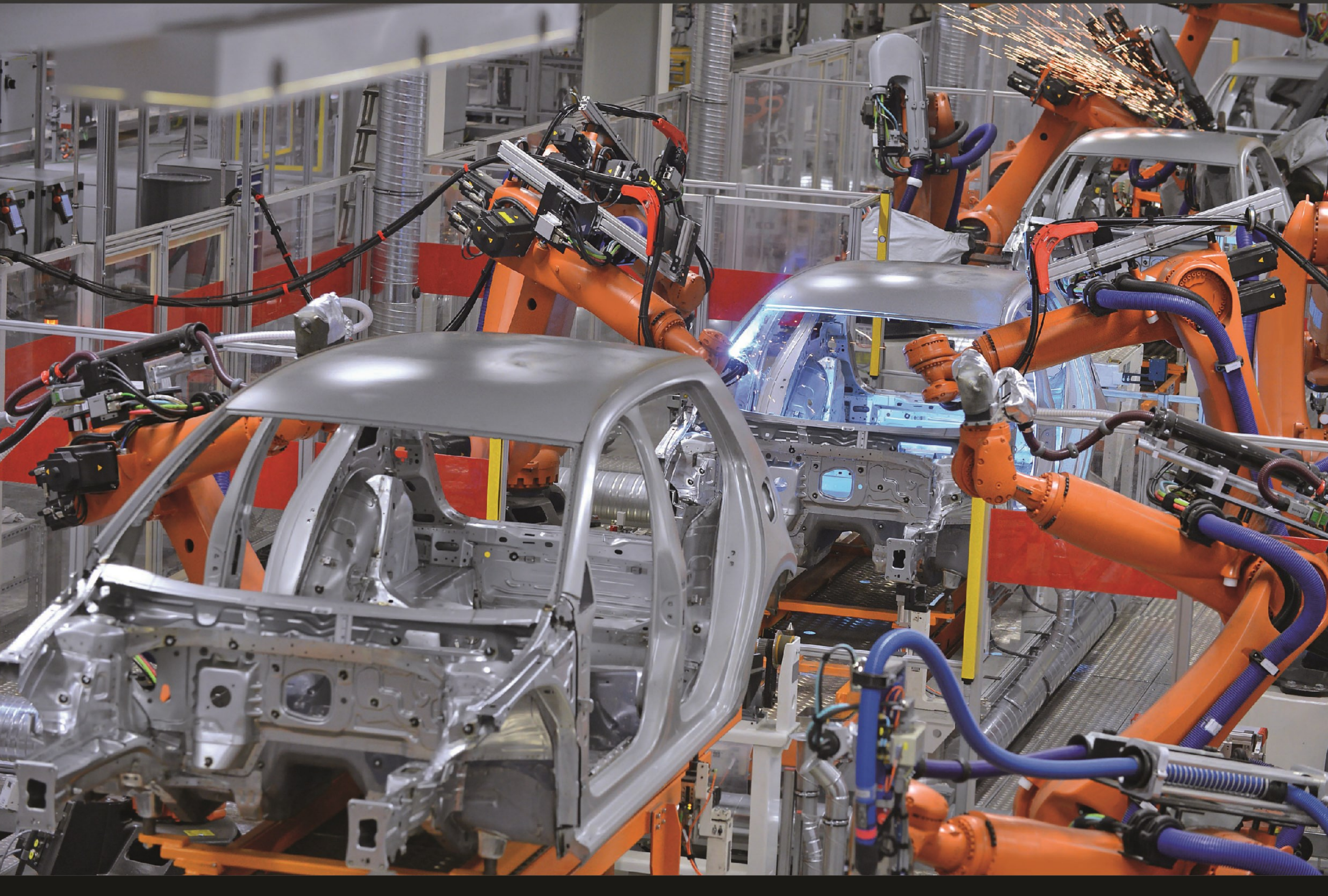
//ALLCODE™

ROBOT ARM PRODUCTION CELL

Robot arm Development II



MATRIX

CP5390

www.matrixtsl.com

Contents

Getting started

Before you start the worksheets you will need to go through a number of steps in understanding the hardware and software that you will be using.

Firstly please read through the information in the Reference section. You don't have to absorb all the information here: just make sure that you know what is there and where it is so that later on you know where to look up the relevant information.

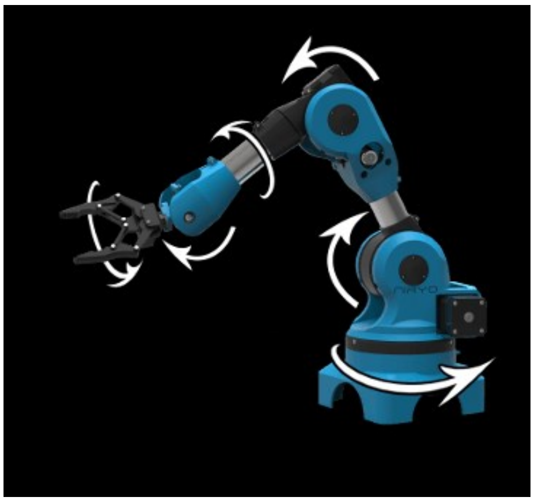
If you are a teacher please read through the Teacher's notes section. There are many teaching options with this robot arm and this section will guide you through the options, tell you how we feel you should be using the product for teaching, and give you an understanding of the learning objectives and learning difficulties of each worksheet.

You will need to make a decision on how you want to program the robot arm. You can use PC with the Windows apps provided, LabView, C++, or any other language. You can also use a Raspberry Pi with a number of languages or you can use mobile phone with Appinventor.

Once you are familiar with the hardware and software then you can work through the worksheets provided. These are language independent and are structured to build on your learning step by step.

Worksheet 1

Investigating the workspace



Robot arms have a number of axes. This determines their range of movement and their capability.

Most robot arms have base rotation, shoulder rotation and elbow rotation. This ,and the length of the limbs of the arm, determine the reach of the arm.
The arm movement is confined in a three dimensional area known as the 'workspace'.

Over to you:

1) Go through the following information in the Reference section:

- Understanding the robot arm
- Assembly instructions
- Block diagram
- PC USB driver and software set up
- Basic control software app

2) On your PC load the BasicControl.bat software.

3) Press the PLAY button on the top left.

4) Explore the range of movement with the Base slider, the Shoulder slider and the Elbow slider. The units here are in stepper motor steps.

5) Using the gripper pick up one of the counters.

6) Using the Base slider, the Shoulder slider and the Elbow slider explore the XY limits of the work-space where the counter can be picked up and placed down. This is also marked out on your mat.

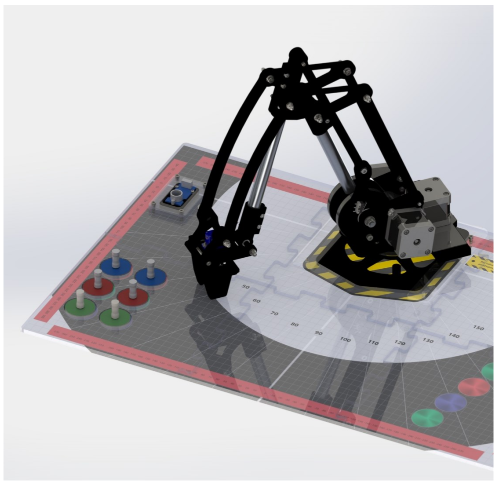
7) Use the position control and save/load controls to store and recall a sequence of three movements defining the extent of the workspace - locations A to E on the diagram.

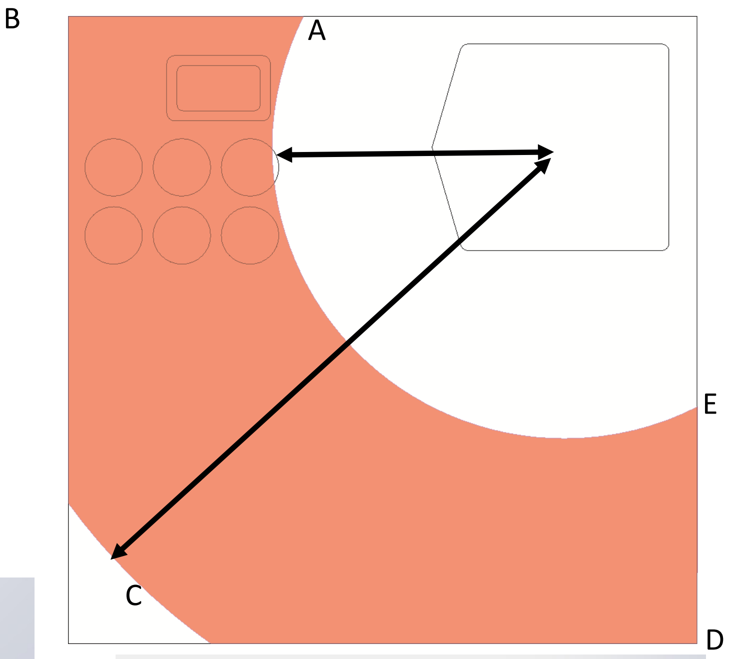
8) With the arm at 90 degrees what is the height of the workspace?

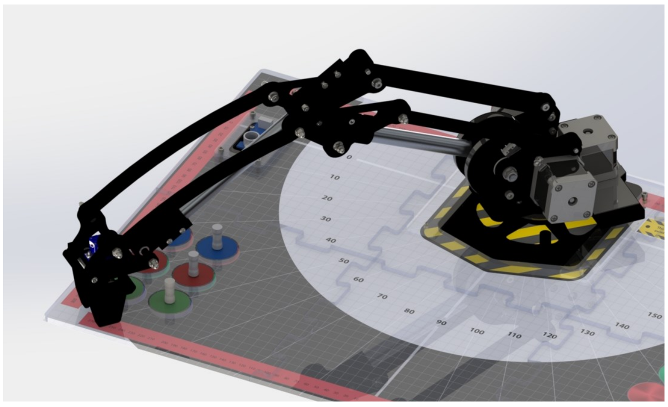
9) For two points of the arm - horizontal to the top of the workspace and 45 degrees to it fill in the table on the following page.

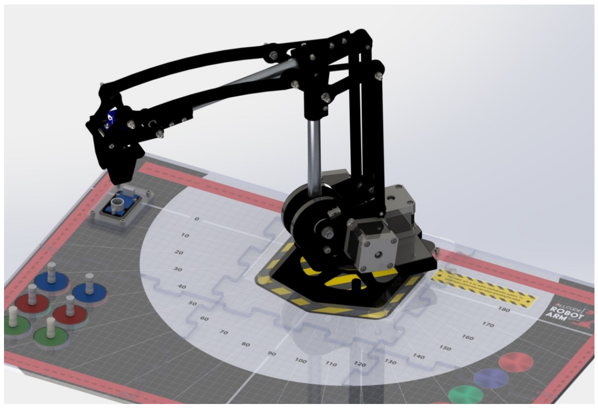
Worksheet 1

Investigating the workspace


Closest to base




Furthest from base


90 degrees

	Base angle	Base steps	Shoulder angle	Shoulder steps	Elbow angle	Elbow steps	X	Y	Z
Closest									
Furthest									
Closest arm at 90									

So what?

The workspace of a robot is limited by the geometry of the robot arms. When designing a robot cell one of the first things you will need to look at will be the dimensions of the workspace and the capability of the arm. The fundamental unit of motion is a stepper motor step. This produces a rotation in the arm in degrees for each axis. This in turn positions the arm in the X, Y, Z plane where it can be described in mm. The mathematics involved in describing the relationship between these three co-ordinate systems and calculating them is called 'Kinematics'.

Worksheet 2

Pendant programming

You can program a robot arm in several ways. The most common methods are to write a programme in some kind of programming language or to 'teach' the robot on the factory floor. When teaching a robot we often make use of a 'pendant' which connects to a PC or to the robot itself. In this worksheet you will use a PC based app with a simulated pendant.

Image shows a robot arm with teaching pendant.

Over to you:

1) Go through the following information in the Reference section:

- Pendant program
- G code reference
- Colour sensor

2) On your PC load the Pendant.bat software.

3) Press the PLAY button on the top right.

4) Make sure that the workspace is set up with the colour sensor in place and with it connected to the arm, and that the 6 coloured counters are in place.

5) Start off by understanding how to move the arm in X, Y, and Z.

6) Next understand how to build small G code programs.

7) Now develop a program using the Pendant program to pick up one counter and put it on the appropriately coloured circle on the mat.

8) Expand your program so that it then picks a counter up again and places it on the relevant coloured circle

9) So what?

It can take time, but teaching robots using a pendant produces great results.
Good preparation of the workspace in terms of understanding the positions of the workpieces, and the robot gripper location can save a great deal of time.

Additional tasks

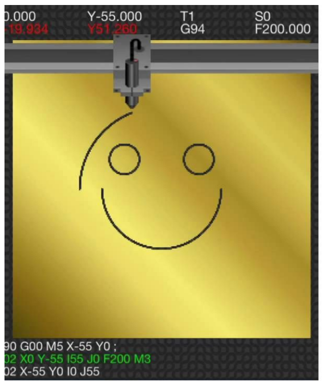
Two characteristics or robot arms are often quoted: accuracy and repeatability.

- Accuracy is how closely an arm can move to a designated XYZ position.
- Repeatability is the accuracy of an arm when moving to a particular position.

Investigate the arm properties and come up with figures for accuracy and repeatability.

Worksheet 3

G code programming



```
.000        Y-55.000    T1     S0
19.934      Y51.260     G94    F200.000
```

```
90 G00 M5 X-55 Y0 ;
02 X0 Y-55 I55 J0 F200 M3
02 X-55 Y0 I0 J55
```

```
G00 X-25 Y20
G91 X-10
G02 X10 Y-10 I10 J0 F200 M3
G02 X-10 Y10 I0 J10;
G90 G00 M5 X0 Y0;
G00 X25 Y20
G91 X-10
G02 X10 Y-10 I10 J0 F200M3
G02 X-10 Y10 I0 J10;
G90 G00 M5 X0 Y0;
G00 X40 Y0
G02 X-40 Y0 I-40 J0 F200 M3
G90 G00 M5 X-55 Y0;
G02 X0 Y-55 I55 J0 F200 M3
G02 X-55 Y0 I0 J55
```

G codes are used extensively in manufacturing engineering to describe the movement of machines and the actions of the tools on them. G code consists of a standard set of commands and Machine specific commands - also called M codes.

Image shows a G code program to draw a smiley face using a pen plotter.

Over to you:

1) Go through the following information in the Reference section:

- G code reference

2) Make sure you understand how to use the Pendant software to load and save simple G code files.

3) Develop a program using the Pendant program to pick up a red counter and put it on the appropriately coloured circle on the mat.

4) Now open the program in Notepad and use copy and paste to duplicate it. Calculate the changes in dimensions in X and Y for a green counter and edit the program accordingly. Save the program.

5) Open the program in the Pendant software and test that it works.

6) Using Notepad expand your program so that all of the counters are removed from their starting positions and placed on the mat in the appropriately coloured circles.

So what?

Pendant programming can be great when only one simple operation is needed. But when an action applies to many different items in a workspace then sometimes its quicker to edit a program numerically rather than teach the robot.

There are many common G codes for programming robot movement. In practice the G and M codes will vary from one manufacturer to another.

Additional tasks

Alter your program so that it reverses the placement and returns the counters to their starting locations.

- What are the problems with this?
- How would you improve the mechanics of the robot arm to alleviate these problems?

Worksheet 4

Starting API programming

Pendant programming and G code style programming are great tools for simple programming of robots. But in situations where further robot intelligence is needed, or where the robot has to interface to other control systems, then you will need to utilise some kind of programming tool.

Image shows a bomb disposal robot with robot arm.

Over to you:

1) In the reference section you will find a some examples on API programming using Flowcode, RPi / python, Matlab and other languages. Once you have made your choice familiarise yourself with the programming language requirements and the API.

2) You will also need to understand how you are going to communicate with the robot arm and you may need to configure the robot arm Bluetooth or IP address using the Configure software application. See the reference section for details.

3) Print out a copy of the API reference sheet which shows you the available commands.

4) For your first program make sure that you can make use of two commands: Home and Setmotor. The pseudo code of the program you will construct is as follows:

```
1.   Open RA library
2.   Open COM port
3.   Home all
4.   Setmotor (1,0,12)
5.   Close COM port
```

Line 1: Open the Robot Arm library
Line 2: Open communication to the robot arm
Line 3: Reset robot to home position
Line 4: Set motor 1 to coordinates:
    Least Significant Byte 0
    Most Significant Byte 12
Line 5: Close communications to the robot arm

So what?
This is your first program. Once you have this up and running then you can simply refer to the API reference and build on your program to complete the tasks in the following worksheets.

Worksheet 5

Simple production

Robots are great at repetitive tasks that require high precision. Very often in production environments workpieces moved from one machine to another by a robot with storage systems between the machines. Workpieces are often not moved from and to the same place and some intelligence needs to be used to determine where to pick and place a workpiece.

Image shows a production line robot which moves a workpiece into and out of a CNC machine

Over to you:

1) Using the Basic Control app find out the following:

- First red counter storage position: X1, Y1, Z1
- First red counter mat position: X2, Y2, Z2
- Gripper setting which just gives good purchase on a counter spindle: A.

2) Write the following program:

```
1.  Open RA library
2.  Open COM port
3.  Home all
4.  SetMovementSpeed(0,10)
5.  MoveToolXYZ (X1, Y1, Z1)
6.  SetGripper (A)
7.  MoveToolXYZ (X2, Y2, Z2)
8.  SetGripper (0)
9.  Close COM port
```

2) Check this works well.

3) Using the dimensions of the counter storage slots and the counter destination slots on the mat: expand your program so that all the counters are picked up from their storage slots and placed on the mat in the appropriate place.

So what?
Now you should see one of the benefits of using a fully fledged programming language: replicating simple routines for many components becomes easier.

Worksheet 6

Sorting

Robots start to become really useful when they are fitted with sensors which give them information about the objects that they are handling. Then the robots can start to make intelligent decisions.

Image shows robots of the future recycling rubbish.

Over to you:

1) Use the Basic Control software app to explore the colour sensor. Just using your fingers move different coloured counters over the sensor and make a note of the values in Red, Green and Blue for different colour counters.

2) Take your program from the previous worksheet and modify it so that it tests each counter with the colour sensor before placing it.

```
GetColourSensor (R,G,B)
```

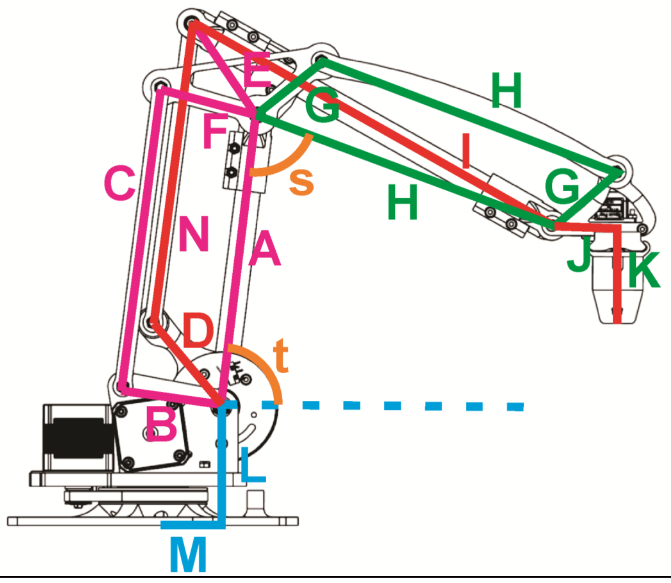
3) Test your program by mixing up the counters in their slots and checking that the robot arm is able to sort and place them in the right place on the mat.

So what?

This simple program starts to give you an idea of the power of robotics when the robot can start to make decisions. In factories robots with vision systems now carry out complex movements and decision making.

Worksheet 7

Kinematics calculations



The drawing on the left shows the geometry of the robot arm. The API inside the robot uses this geometry and Kinematics to convert the rotational motor of each of the three stepper motors into X, Y, Z co-ordinates.

In this section you can check your understanding of Kinematics.

Over to you:

1) Your first task is to make sure that you can convert a stepper command in steps into angles for each of the joints. To start with just focus on the Base using the SetMotor API command. Write a program that converts Base motor steps to an angle. The angles are marked clearly in the mat. Use this to check your working.

2) Your second task is to repeat this task to make sure that you can predict the angle of the shoulder to the mat. You will need to devise some kind of large protractor to help you measure the angle.

3) Lastly you can repeat the exercise for the Elbow.

4) Use Kinematics to take predict the position of the tool to X, Y, Z data. Use the SetMotor API command to move to a position in steps and then use the MoveToXYZ command to check your calculations against the API inside the robot arm.

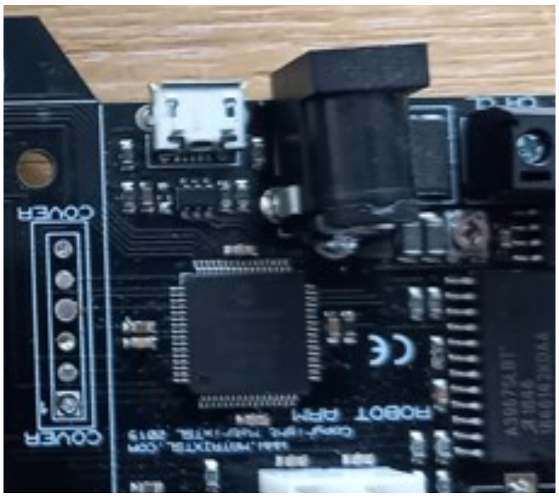
Data you will need:

A = C = lower arm = 176mm
N = second gear pivot arm = 180.34mm
B = F = side pivot points = 60mm
D = 2nd gear arm centre to bearing = 65mm
E = arm socket to pivot bearing = 67.33mm
G = pivot bearing to bearing = 50mm
H = upper brace = 188.30mm
I= upper arm = 247mm
J = bearing to head centre = 39.15mm
K = head height = 59.55mm
L= shoulder height (not acrylic) = 71mm
M = central base axis to shoulder centre = 41mm
s= upper and lower arm angle
t = base to lower arm angle

(note that s and t are not the angles in the Basic Control App: these are angles from zero switches.)
Output angle per step:@ 0.044

Worksheet 8

Microcontroller programming



Robot arms make use of an Application Programming Interface software to allow easy reprogramming. You have seen the benefits of this in the previous worksheets. They don't have to work that way: robotic devices can be configured not to be reprogrammable and to only carry out one function. In this section you get a chance to design such a programme at the microcontroller level.

The photograph shows a clos up of the dsPIC microcontroller on the robot arm.

Over to you:

1) Go through the following information in the Reference section:

- Microcontroller connections
- Reinstalling the API

2) Using any language of your choice develop a program to flash one of the LEDs. Send this to the robot arm using the bootloader. This is described in the section on 'reinstalling teh API'. You now know that you are able to write and transfer a program to the robot arm. If you are using Flow-code as a programming environment you will find that the latest version has all the components and routines you need for the robot arm.

3) Expand your program so that you can move each of the stepper motors. Calibrate the stepper motors (you might want to use the BASIC Control.bat app to help you here) so that you can give the routines an angle for each stepper.

Activate the gripper and read the colour sensor (use each of the three A,B, C LEDs as the output of your colour sort routine).

2) Build on this program to pick, sort and place one of the counters. Test your program with different coloured counters.

3) Build on this program to make a program that sorts all coloured counters.

Teacher's notes

Teacher's notes

About this course

This document has two functions: it is both a datasheet on the Allcode robot arm, and it gives ideas for activities. There re three types of student - and three corresponding activities and learning opportunities - that can be undertaken with the arm:

Technician level
At this level we want technicians to learn about robot arms and robot arm workspaces. We want them to understand the advantages of working with robots and to understand their limitations. We want them to plan simple robot activities to prepare them for using real commercial robots in the workplace. We draw a line with these students at full programming.

Production engineer level
At this level we want students to start to use high level programming languages for programming robot workcells. The use of programming languages allows more functional workcells to be developed and has huge advantages when designing more complex production lines.

The Allcode robot allows students to use a wide range of programming languages and platforms: from App Inventor on a mobile phone to LabView on a PC.

Robot designer level
At this level we want to give students who are interested in designing mechatronic systems at a low level the opportunity to spend some time programming the robot arm to produce base level functions that high level programming languages can take advantage of. There are two types of student who we are interested at this level:

Kinematics: these students of robotic will want to make sure that they understand the geometry calculations needed to convert the rotational motion of motors into the linear coordinates used by Production Engineers. The Allode robot API supports these students really well as the students can send commands in both motor steps and motor X,Y,Z commands.

Microcontroller programming: these students - mostly interested in electronic system development - can reprogram the 16 bit microcontroller on the robot arm with their own program - effectively their own API. This allows them to study the creation of microcontroller programs in a motivation electromechanical environment.

Within all these levels it has to be stressed that this document is not a full teaching course - it gives a structure of worksheets that students can use with the support of a teacher.

Syllabuses
The robot arm is suitable for use with a wide range of syllabuses including:
BTEC National
Unit 6 - microcontroller programming

BTEC Higher National
Unit 6 - Mechatronics
Unit 15 - Automation, Robotics and PLCs
Unit 77 - Industrial robots

Learning Objectives

- Robot construction and geometry
- Workspace planning
- Simple robot arm movement
- Robot arm pendant programming
- Robot arm G code programming
- High level programming with any of: Flowcode, BASIC, LabView, Matlab, Python, App Inventor, C++, C# etc. using API.
- Robotic sensors
- Kinematics and robot geometry
- Microcontroller programming for robotics
- Controlling robots with PLCs using CAN bus (two wire) or MODbus (Wi-fi connection).
- Robot manufacturing cells.

Worksheet	Notes for the Instructor	Time
1	In this worksheet students use a PC all to investigate the fundamentals of robot arm movement. They understand that there are three stepper motors that control the geometry. They also understand that moving the arm using the angle controls is really hard. To get to a position you need to continuously alter the motors to get the arm to where you want it to be. Students will find this a little frustrating - but it will teach them how the arm moves. In this worksheet students also understand the limitations that the arm geometry puts on the workspace and the value of planning the workspace of a robot. Students will also understand that to get to an X, Y, Z location it is not just a matter of adjusting the angles of the steppers and the arm will go to that location: the steppers will need to move in a certain sequence or the two parts of the arm interfere with each other. When the parts of the arm interfere with each other the arm judders. When students first come to the arm the natural conclusion is that the belt is slipping on the teeth of the gears. This is not so. The juddering is the stepper motors slipping. This will not harm the arm. Don't get the students to spend too long here: they will want to quickly get on to programming tasks.	60 mins
2	The Pendant programming app makes the robot arm very accessible to all students. In this worksheet students just familiarise themselves with the Pendant app. For the faster students we have suggested an investigation into accuracy and repeatability. Other tasks can also be set.	60 - 120 mins
3	Having learned how to use the Pendant app in this worksheet students start to solve the problem of moving the counters from the starting position to the final position. They will learn the fundamentals of the G code language and G code programming. They will learn that pendants are useful to a point but that moving to a text based programming system is going to offer greater flexibility. For the faster or more able students we have suggested investigating returning the counters to their positions and the problems that this brings: mainly the design of the gripper and its accuracy and the possible need for location devices on the acrylic mat to ensure counter position is maintained. At this point there are also lots of other exercises you can set for students.	180 mins

Worksheet	Notes for the Instructor	Time
4	You may choose to start your students at this point rather than get them to work through the previous worksheets. This worksheet deals with starting to use a programming language with the robot arm. As the arm has an API with USB, Bluetooth or Wi-fi connectivity the options for programming are very varied. This worksheet is just to flash a LED to make sure the students have communications and all software in place.	30 mins
5	Once students are up and running with the arm and the software they can start to build the basics of a program that will pick up, sort and place the counters. This task just asks them to set up a program to pick up and place the counters. The length of time this will take is clearly very much influenced by the experience of the student.	6 hours
6	Once students have the basics then they can move on to picking a counter up, testing the colour and then placing the counter in the appropriate position. From this point there are lots of other exercises you can set the students to do with the arm - do not feel that you need to just work with the counters and mat we have provided: you can use the arm with other pieces and set other tasks., You can also get students to design electromechanical add-ons for the arm and use the relay on the main circuit board to power other devices.	2 hours
7	Some students will want to understand the geometry calculations that convert an angular rotation of the three stepper motors into movement in the X, Y, Z plane: kinematics. This is complex. Enough dimensional information is given on this worksheet to allow the calculations to take place - but the task ahead of these students is not trivial. The API calls provides with the robot arm include these calculations. Students can check their work with the API calls themselves: they can move to a position, calculate the X, Y, Z of that position and can then compare that with the MOVE(X,Y, Z) API call to see if there is a difference. Ambitions students can also develop routines for inverse kinematics where they calculate stepper motor steps needed for an X, Y, Z position. Time needed here is questionable - this is advanced learning and so dependant on student ability. You can just ask the students to solve this problem. Or you can give them the formulae they need which are: Forward kinematics: Z = F + SIN (T) * A - COS (S - (1.570796 - T)) * D + K OFFSET = SIN (S - (1.570796 - T)) * D + COS (T) * A + G + H X = COS (Base) * OFFSET Y = SIN (Base) * OFFSET	20 hours

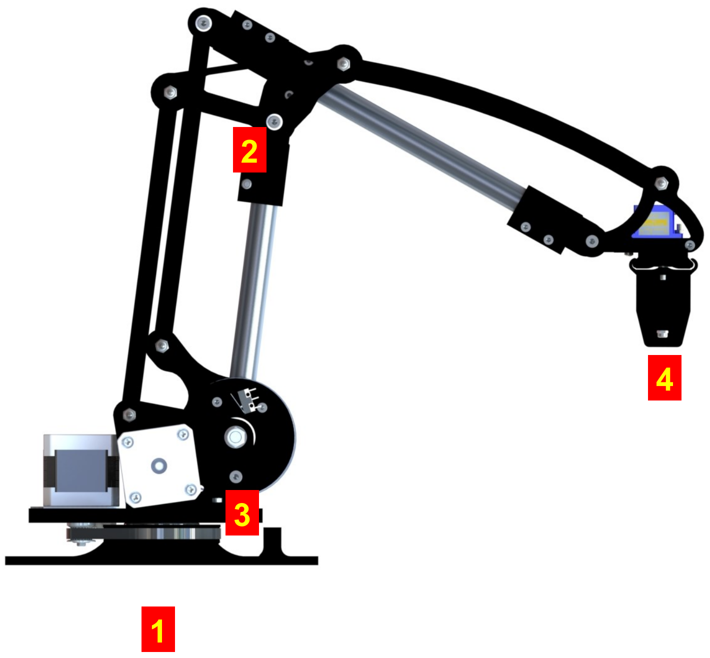
Worksheet	Notes for the Instructor	Time

7 cont	Reverse kinematics:
	Base = ATAN2 (Y, X)
	X = X – COS (base) * (G + H)
	Y = Y – SIN (base) * (G + H)
	Z = Z + K – F
	DIST1 = SQRT ($X^2 + Y^2$)- G
	DIST2 = SQRT ($DIST1^2 + Z^2$)
	CALC1 = Z / DIST2
	CALC2 = ($DIST2^2 + A^2 - D^2$) / (2 * DIST2 * A)
	CALC3 = ($A^2 + D^2 – DIST2^2$) / (2 * A * D)
	T = (ATAN2 (CALC1, SQRT(1 – $CALC1^2$)) + ATAN2(SQRT(1 – $CALC2^2$), CALC2))
	S = ATAN2 (SQRT (1 – $CALC3^2$), CALC3)	

8	Using the robot arm as a microcontroller based application is a key objective behind its development. The bootloader in the arm easily allows new code to be developed for it and the connection chart on the 'Microcontroller connections' page should be enough to allow students to program the hardware. The difficulty here is that the student will be over writing the API in the arm hardware. You will need to take a decision as to whether to get students to replace this when they have finished using it so that the arm is ready for the next user.	30 mins

Reference

Understanding the robot arm



There are four joints in the robot arm: Base, Shoulder, Elbow and gripper. In robotics terms there are 'three degrees of freedom'. This relates to the fact that there are three axes in which the robot arm can move.

The mechanics of the arm are designed to maximise the payload(the amount the arm can lift). This is achieved by placing the heavy motors on the base platform and by using a system of levers and cogs to allow the arm to move with great precision within its range of motion.

The fundamental actuator for each axis is a stepper motor. Each stepper motor is connected to a drive cog and an axis cog. The drive cogs have 20 teeth. The axis cogs have 102 teeth. This gives us an effective gearbox ratio of 5.1:1.
Each stepper motor takes a single step of 1.8 degrees. The driver chips make use of 'microstepping' technology which gives 8 positions withing each 1.8degree increment. This gives an effective step accuracy of 0.044 degrees per step.

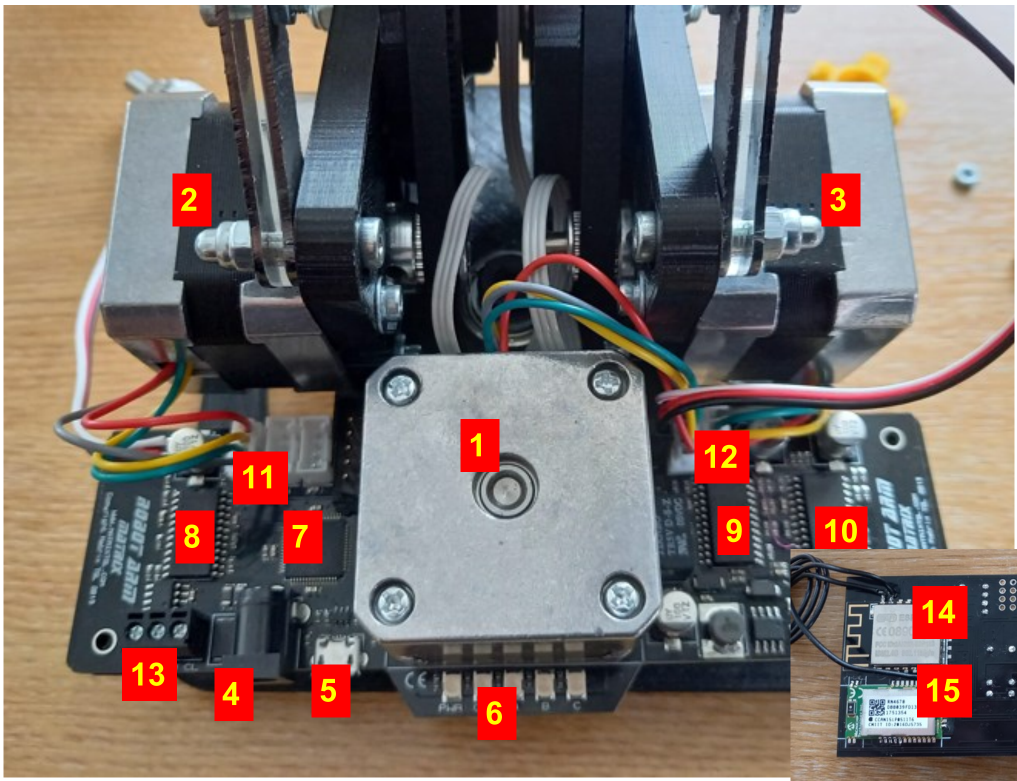
The gripper is comprised of a small servo motor and a mechanical actuator. It is possible to set the position of the server with 255 different positions between fully open and fully closed.

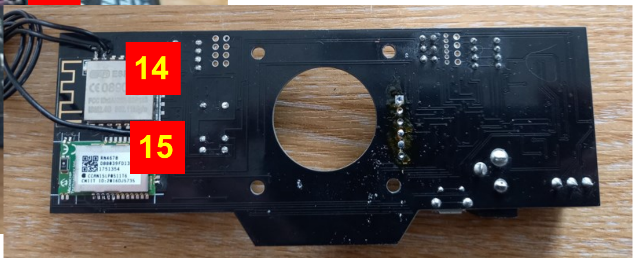
The two main levers are manufactured from aluminium tubing for light weight and strength. The bulk of the mechanics are manufactured from 3D printed plastic.

A single circuit board is used to control the robot arm.

1) Base
2) Elbow
3) Shoulder
4) Gripper

Understanding the robot arm





1) Base stepper motor
2) Elbow stepper motor
3) Shoulder stepper motor
4) 2.1mm power jack connector (12V)
5) USB connector
6) LED status lights:

   PWR: power present
   COM: communications from host
   A: Base movement
   B: Shoulder movement
   C: Elbow movement
7) Microcontroller
8) Elbow motor driver
9) Base motor driver
10)S houlder motor driver
11) Left to right: Elbow stepper connector, no fit connector, colour sensor connector, stepper microswitches —used for detecting 'home' on all three axes, relay connections.
12) Left to right: Gripper servo connector, Base stepper connector, Shoulder stepper connector.
13) CAN bus terminals: Terminate, High, Low. (120R between T and CH for termination)
14) Wi-fi module
15) Bluetooth module

Understanding the robot arm

LED functions on startup
During startup we have the following pattern after the scrolling animation. (Some steps are too fast to see unless there is a problem.)

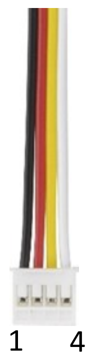
1) C Flashing – USB Comms Starting Up
2) B Flashing – CAN Comms Starting Up
3) C + B Flashing – Bluetooth Comms Starting Up
4) A Flashing – WIFI Comms Starting Up
5) A + C Flashing – Loading Stored Calibration Settings

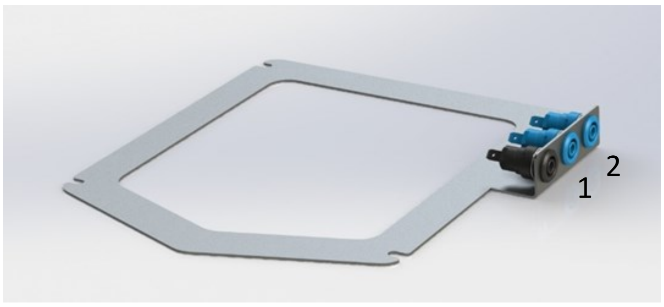
Connections on expansion port

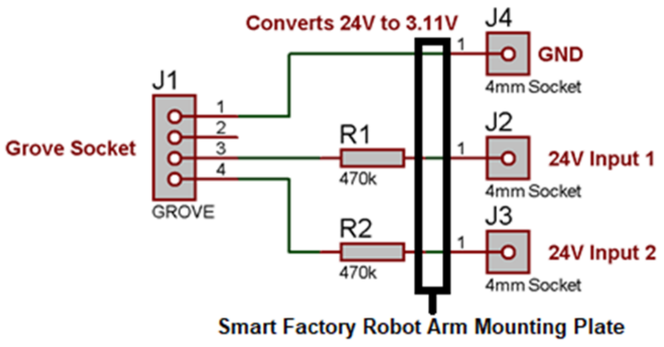
One of the connectors is provided to allow the robot arm to interface to other programmable devices and to use these inputs as a trigger for various actions. The connector is a standard Grove connector.

The connections are as follows:

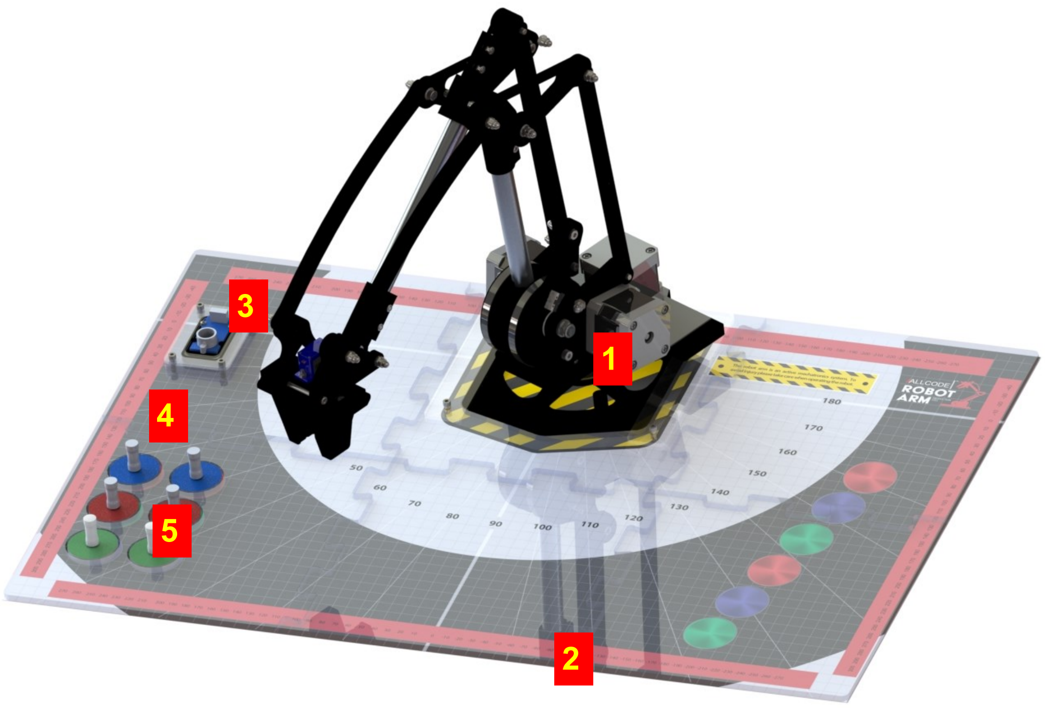
Grove Connector Pinout – 1: GND, 2:VCC, 3: D0 or Input 1, 4: D1 or Input 2







Understanding the robot arm



What's in the package

The robot arm workspace can be seen above. It consists of:

1) Robot arm. Supplied fully assembled and tested. Simply plug power supply in to start.
2) Plastic mat cover: 3mm clear acrylic to provide a location template for arm, counters and sensor
3) Colour sensor: returns Red, Green and Blue readings for any object placed on top of it.
4) Counters with spindle that can be picked and placed.
5) Colour work mat placed underneath the acrylic cover.
6) Power supply and micro USB lead.

Assembly and installation instructions

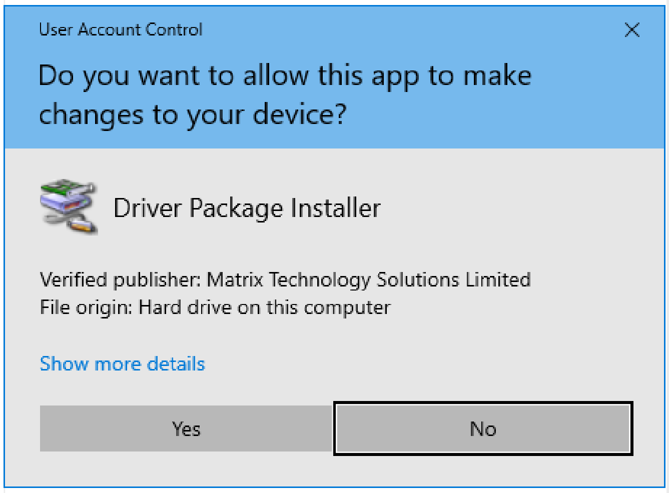
1) A 12V power supply with multiple country mains adaptor plugs is supplied with the arm. Make sure the power supply is configured for 12V output using the small key supplied with the power supply. (see image above right)
2) Insert the power supply into the robot arm power jack. Use A USB to micro USB cable to connect the arm to the PC.
3) Install the robot arm driver by following the instructions on PC driver installation below.
4) Download the robot arm software from our web site. Run the Basic Control software. See the page below on using the Basic Control Software and make sure that all parts of the robot arm function.

PC USB driver and software set up

Before using the robot arm with a PC you will need to install USB driver. This can be downloaded from www.matrixtsl.com along with the basic software applications for using the robot arm.

To install the USB driver you may need administrator permissions.

Run the RobotArmV2 Installation for your machine.



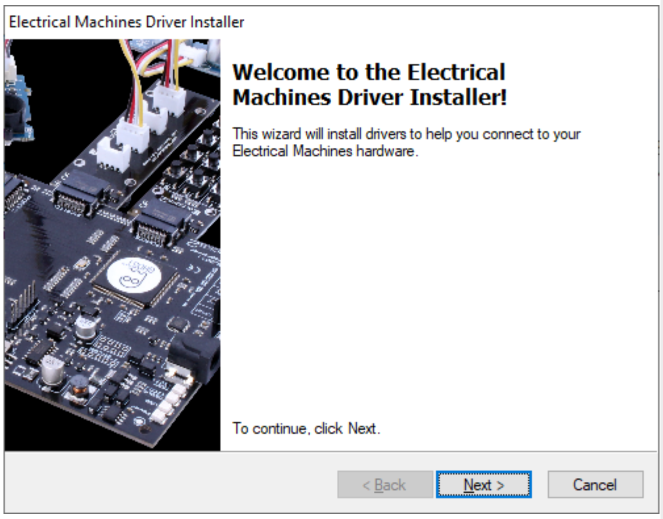
User Account Control

Do you want to allow this app to make changes to your device?

Driver Package Installer

Verified publisher: Matrix Technology Solutions Limited
File origin: Hard drive on this computer

Click Yes



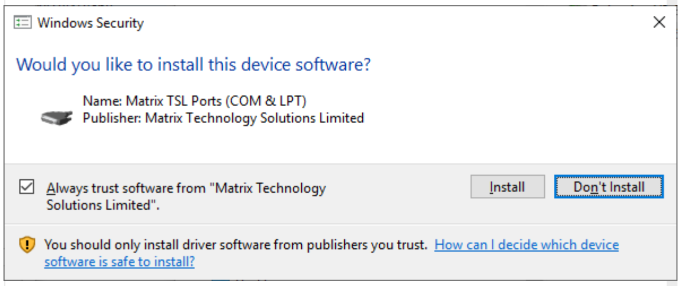
Electrical Machines Driver Installer

Welcome to the Electrical Machines Driver Installer!

This wizard will install drivers to help you connect to your Electrical Machines hardware.

To continue, click Next.

Click Next



Windows Security

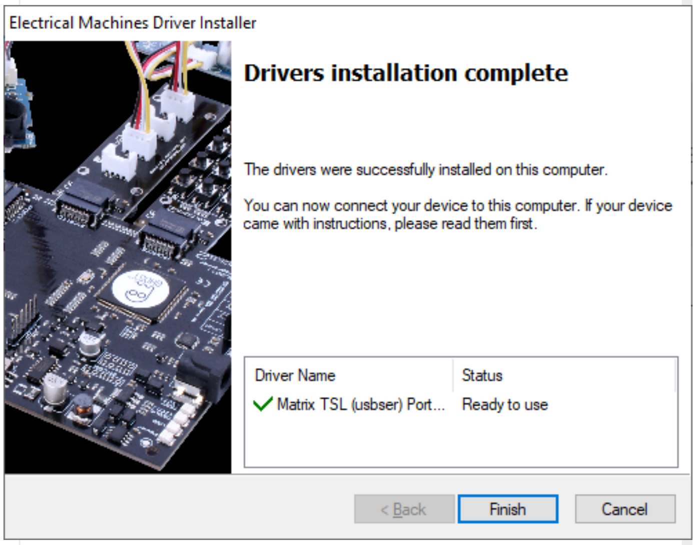
Would you like to install this device software?

Name: Matrix TSL Ports (COM & LPT)
Publisher: Matrix Technology Solutions Limited

Always trust software from "Matrix Technology Solutions Limited".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

Click Install



Electrical Machines Driver Installer

Drivers installation complete

The drivers were successfully installed on this computer.

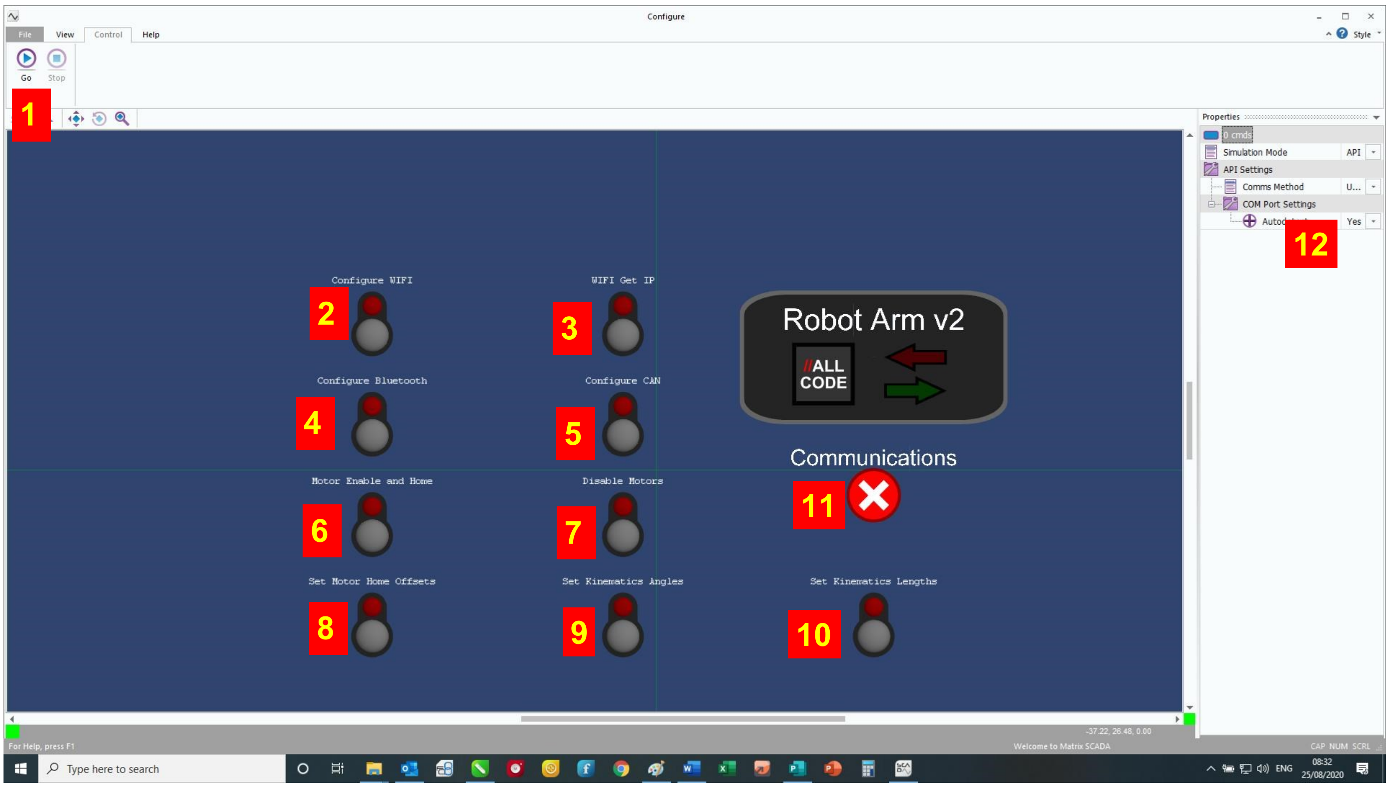
You can now connect your device to this computer. If your device came with instructions, please read them first.

Driver Name	Status
✓ Matrix TSL (usbser) Port...	Ready to use

Click Finish.

The USB driver is now installed and ready to be used.

Configuration software



Configuration should be carried our via USB.

1) STOP/PLAY controls

2) WIFI config—Enables or Disables WIFI functionality, default Disabled.

3) WIFI Get IP—Gets the IP address of the WIFI.

4) BT config—Enables or Disables Bluetooth functionality, default Enabled.

5) CAN Config—Enables or Disables CAN bus functionality, default Disabled.

6) Enable and motor Home—Homes the motors to their offset position.

7) Disable Motors—Disables the motors so they are no longer active and drawing power.

8) Set Motor Home Offsets—Sets the offset step count from the limit switch home position for base rotation. This is calibrated in the factory.

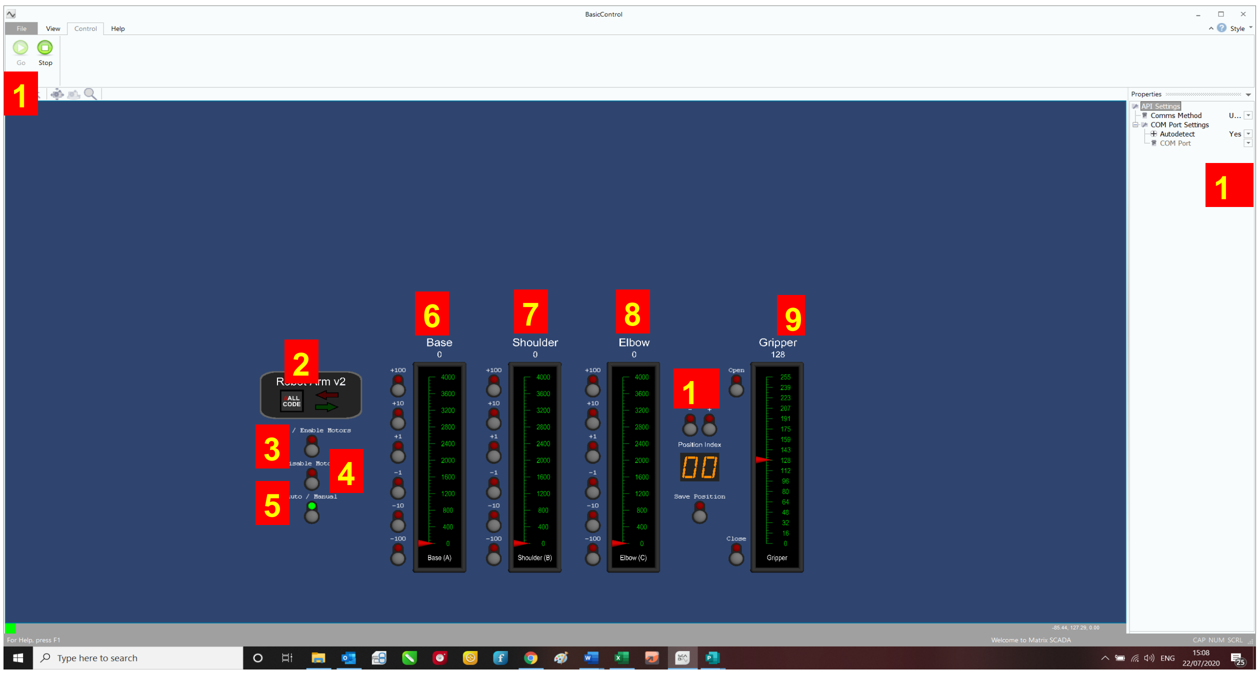
9) Set kinematics angles - after using the measurement jig and Open BasicControl.bat you can enter the shoulder and elbow angle to fine tune accuracy of the arm. This is calibrated in the factory.

10)Set Kinematics lengths- allows you to enter exact values for lower and upper arm lengths - A and H respectively.

11) Communications flag

12) Communications set up - USB recommended for configure.

Basic control software



This basic level software is supplied with your robot arm package. You can download this program from our web site. To run the program simply Run 'BasicControl.bat'.

1) STOP/PLAY controls

2) Comms window: green arrow flashes when command sent to arm, red light flashed when data received from arm.

3) Home switch: sends arm to the home position.

4) Enable/Disable motors

5) Auto/Manual switch: when in Auto the arm is moved as soon as sliders are adjusted. When in Manual the arm is only moved on request.

6) Base slider and switches: adjustment in steps.

7) Shoulder slider and switches: adjustment in steps.

8) Elbow slider and switches: adjustment in steps.

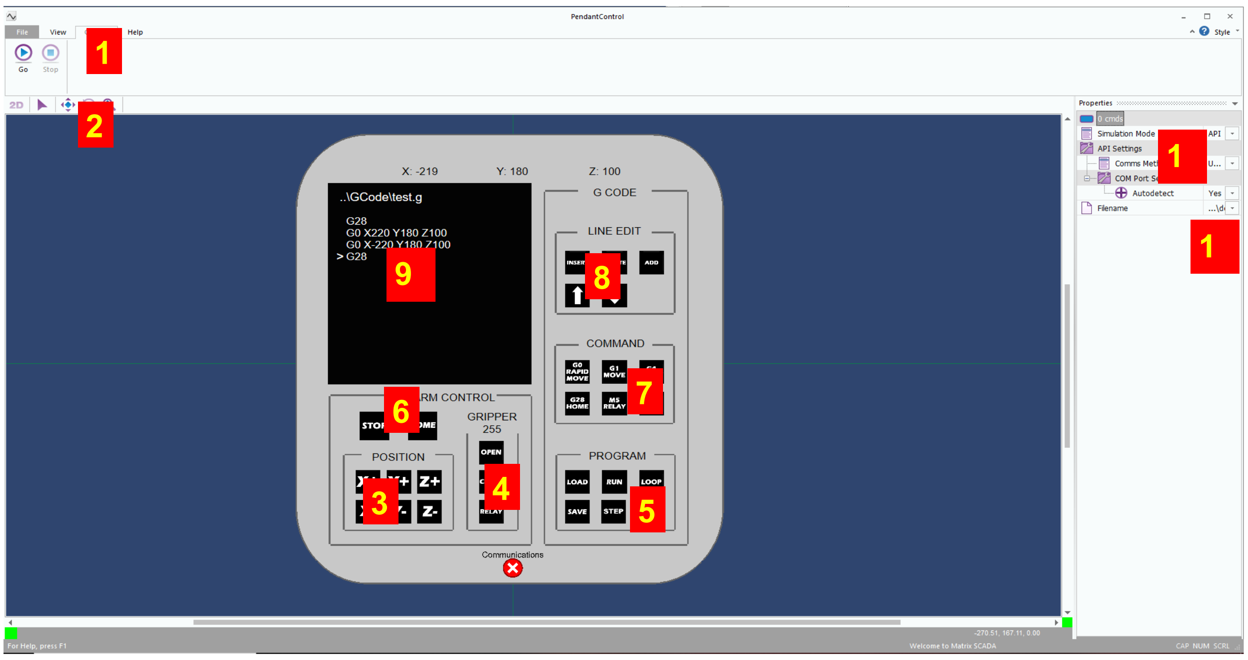
9) Gripper servo motor adjustment

10) Position control and save/load controls: allows users to store up to 20 different positions and recall them.

System set up options including USB, Bluetooth, Wi-fi and CAN bus.

If using Wi-Fi you will need to enter the IP address of the arm (see Wi-Fi set up page), the Network Interface (1 to 4: choose the one that gives non zero results) and the port you set.

Pendant control software



```
..\GCode\test.g

G28
G0 X220 Y180 Z100
G0 X-220 Y180 Z100
> G28
```

The Pendant program is designed to replicate pendant style robot programming tools used in industry and give students some experience of programming the system using this style of programming.

The controls on the software are as follows:

1) PLAY/STOP controls - these start and stop the program

2) 2 Zoom and move - moves the controls in the programme workspace

3) X, Y, Z controls for tool

4) Gripper control

5) Program load, save, step and run control

6) Stop and home buttons

7) Command buttons

8) G code line editing controls

9) G code program

10) Comms set up controls

11) Filename control

Note that it is possible to save a program, edit it with Notepad, and then re-load it and execute it.

If using Wi-Fi you will need to enter the IP address of the arm (see Wi-Fi set up page), the Network Interface (1 to 4: choose the one that gives non zero results) and the port you set.

G code reference

The Pendant program makes use of a variant of G code. G codes are used extensively in manufacturing engineering to describe how robots and machines move and act.

The Pendant program allows you to save and load simple G code programs. You can either use the Pendant program to teach the robot and generate G code at the same time, or you can write G code programs in a simple text editor and run them in the Pendant app.

The commands used in the Pendant program are:

G0 - Rapid Move
A movement to an XYZ location at the predefined speed of 8000.
Example
G0 X140 Y0 Z10

G1 - Controlled Move
A movement to an XYZ location at a specified speed ranging from 0 (Slow) to 10000 (Fast).

Example
G1 X140 Y0 Z10 F2000

G4 - Delay
A delay specified in milliseconds
Example
G4 P2000

G28 - Home Motors
Home all motors back to their known origin location.
Example
G28

M5 - Relay
Controls the relay on/off that operates the pneumatic suction.
Example
M5 A1

M6 - Gripper
Controls the gripper servo motor that operates the gripper ranging from 0 (Closed) to 255 (Open).
Example
M6 A128

GOTO #
Goes to a line number in the program
Example
GOTO 2

INPUT 1
Goes to a line number in the program if input 1 is high.
Example
IF INPUT1 GOTO 4

INPUT 2
Goes to a line number in the program if input 2 is high.
Example
IF INPUT2 GOTO 20

Example program

Here is a simple example G Code program that homes the motors and then performs a simple movement.

```
G28
G0 X140 Y0 Z81
M6 A255
G4 P2500
G1 X180 Y20 Z50 F2000
M6 A100
G4 P2500
G1 X140 Y0 Z81 F2000
M6 A255
G4 P2500
```

Understanding the API

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. 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.

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 V2, offers the same platform and language independence as the TV example described above. The difference is the transmission medium for the robot is USB, Bluetooth, WIFI or CAN rather than an infrared beam. This means providing you have a USB, Bluetooth, WIFI or CAN 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.

Understanding the API

As the API reacts to a simple byte-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.

':'	'A'	...	'\n'	...
Command Start	Command Code	Parameters	Command End	Returns

Every command starts off with a ':' character, ASCII value 58 which is then followed by a command code byte that identifies what the robot should do. This may be followed by one or more parameters depending on the command issued. The command is completed with a '\n' characters, ASCII value 10 which executes the command.

For example to change the relay status you might send the following bytes:

```
':', 'E', <value>, '\n'
```

The numeric byte values for this would look like this.

```
58, 69, <value>, 10
```

Where <value> is a 1 or a 0 for ON or OFF respectively.

It should be noted that the API commands for a particular language might have some subtle differences. For example, Python will use something like "`RA2.SetRelay(1)`" whereas C# would look like "`RA_DLL.RA_SetRelay(1);`" and for App Inventor, Flowcode and LabView the appropriate icon would be selected.

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

```
':', 'I', <motor>, <LSB>, <MSB>, '\n'
```

The parameter labelled <motor> can take a value between 0 and 2 to define which motor to move. The parameters labelled <LSB> and <MSB> can each take a value between 0 and 255 to define the step the motor should move to. MSB is effectively multiplied by 256 and added to LSB to make the final step position.

Understanding the API

Below is a table showing the Application Programming Interface specification. This shows you the structure of each command. Later on in this manual you will see examples of each language style so that you can understand how the commands are structured in each language on each platform.

	Cmd Code	Param Bytes	Param	Return Bytes	Return	Details
Get API Version	A	0		1	Version	Gets the API version Number
Home All	B	0		1	Done	Enables all motors and homes all Axis - Returns 1 when
Home Axis	C	1	Axis	1	Done	Enables and homes single Axis 0-2 - Returns 1 when
Get Current Position	D	0		6	A (LSB) (MSB) B (LSB) (MSB) C (LSB) (MSB)	Gets the current step position
Set Relay	E	1	State			Sets the state of the Vacuum relay if connected
Store Current Position	F	1	Index	0		Stores the current position into the specified location
Store Coords	G	7	Index A (LSB) (MSB) B (LSB) (MSB) C (LSB) (MSB)	0		Stores the specified coordinates into the specified location index
Goto Position	H	1	Index	0		Moves to the position stored in the specified location
Set Motor	I	3	Motor Coord (LSB) (MSB)	0		Sets the position of a single motor
Set Motors	J	6	A (LSB) (MSB) B (LSB) (MSB) C (LSB) (MSB)	0		Sets the position of all motors
IsMoving	K	0		1	Moving	Checks to see if any of the motors are moving
GetToolXYZ	L	0		6	X (LSB) (MSB) Y (LSB) (MSB) Z (LSB) (MSB)	Gets the current Tool Position in X, Y, Z coordinates
MoveToXYZ	M	6	X (LSB) (MSB) Y (LSB) (MSB) Z (LSB) (MSB)	0		Moves the tool to the selected X, Y, Z coordinates
Set Gripper	N	1	Position			Controls the position of the gripper servo motor 0-255
Get Colour Sensor	O	0		3	R G B	Samples the colour sensor if connected and returns the reading as RGB values
Set Movement Speed	P	2	Speed (LSB) (MSB) Increment (LSB) (MSB)	0		Configures the movement speed and ramping increment Speed Range 0 (Slow) to 10000 (Fast), 8000 (Default),
Set Wifi Mode	Q	1	Mode	0		Configures whether to host or join a network 0=Off /
Set Wifi SSID	R	STR	STRING	0		Set the WIFI SSID - Max 32 Chars
Set Wifi Password	S	STR	STRING	0		Set the Password - Empty string leaves network open -
Set Bluetooth Mode	T	1	Mode	0		Configures whether to power up BT module 0=Off /
Set Bluetooth Name	U	STR	STRING	0		Set the BT Friendly Name
Set Bluetooth Key	V	STR	STRING	0		Set the BT Passkey
Set CAN ID	W	2	ID (LSB) (MSB)	0		Sets the CAN bus listen ID
Set CAN Mode	X	1	Mode	0		Configures whether to power up CAN module 0=Off /
Get Wifi IP	Y	0		STR	STRING	Collects the WIFI IP address if the WIFI is enabled
Disable All Motors	Z	0		0		Switches off all motor outputs
Write LED	a	2	Index State	0		Writes to a single LED on the Robot Arm 0-4 and sets the state 0-1
Restore Auto LEDs	b	0		0		Restores the default automatic functionality to the LEDs
Reset To Boot	z	0		0		Reset to Bootloader Mode

Reinstalling the API

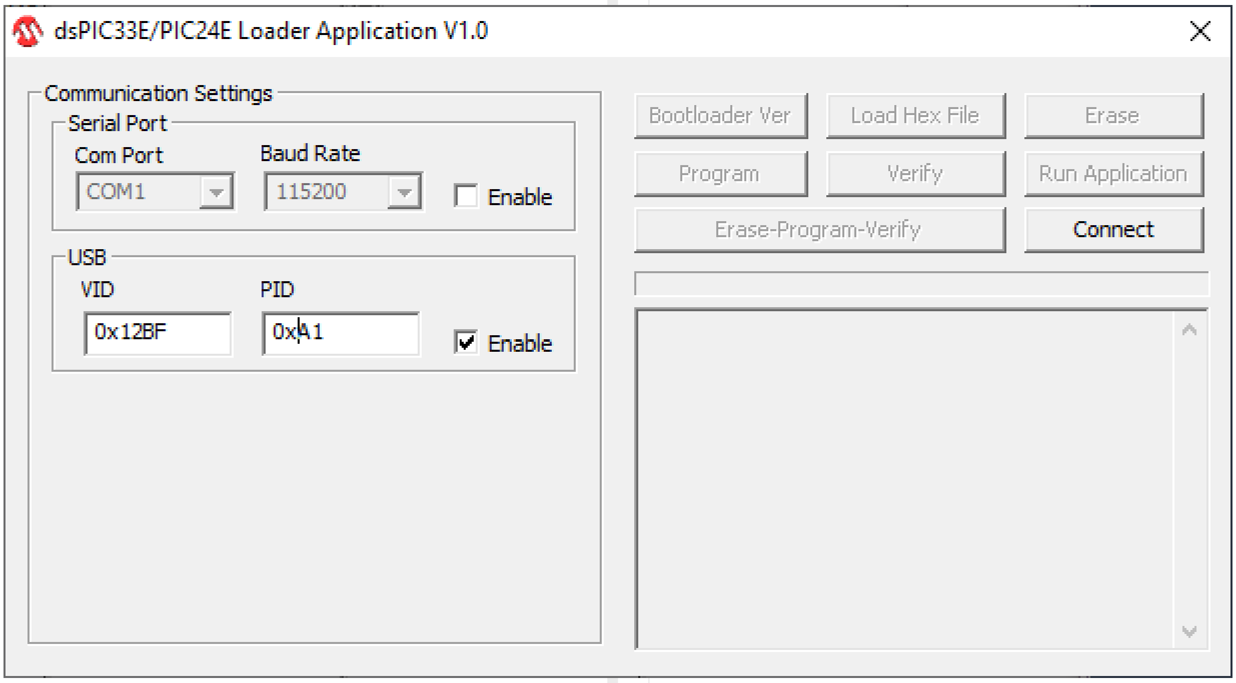
The Application Programming Interface for the robot arm is written in Flowcode and compiled to hex code. The microcontroller on the arm is a powerful 16 bit dsPIC microcontroller. This dsPIC contains a 'bootloader' program. This program resides permanently on the dsPIC. It communicates to a PC via USB and allows the hex code - or 'Firmware' - to be changed easily without the use of any special programming adaptor.

The default program in the robot arm is the Application Programming Interface. If any new firmware has been downloaded to the microcontroller then the API will have been over written. This means that you will need to restore it for the next user.

The process of restoring the API is exactly the same as the process for sending a hex program to the microcontroller.

To restore the API or change the firmware in the arm please follow the instructions below.

1) Run the PC_Loader application from the resource files from the MatrixTSL.com website.
2) Click the Enable button in the USB section.
3) Change the VID to 0x12BF
4) Change the PID to 0xA1
5) Unplug the USB cable from the Robot Arm
6) Power up the robot arm by connecting the power supply to the DC socket.
7) Connect the USB cable to the Robot Arm.
8) All LEDs should go out and the Comms LED should flash to indicate we are in bootloader mode. This should last for approx. 5 seconds before reverting back into normal running mode.
9) Click the Connect button in the PC_Loader application.
10) If the PC_Loader failed to connect go back to step 5 and try again.
11) Click the Load Hex File button
12) Select the hex file to send, the API firmware is named RobotArm2FW.hex
13) Click the Erase-Program-Verify button
14) Click the Run Application button to end the bootloader mode and run the new firmware.



Communication Settings
Serial Port
Com Port: COM1
Baud Rate: 115200
[ ] Enable
USB
VID: 0x12BF
PID: 0xA1
[✓] Enable

CAN bus protocol

The CAN bus protocol is the same as for the other communications methods in that you send the same API bytes and get the same return bytes. The only difference is that a CAN packet can only contain up to eight data bytes.

The default listen ID for CAN commands is 0x100 and so an example command to control the relay might look like this.

ID: 0x100
Byte0: ':'
Byte1: 'E'
Byte2: <value>
Byte3: '\n'

If the API command has return values then these will be sent back from the Arm using the same CAN ID.

Some commands might not fit inside a single CAN packet and may need to be sent across multiple packets.

For example the Store Coordinates command is made up of 10 bytes in total and might look something like this.

ID: 0x100
Byte0: ':'
Byte1: 'G'
Byte2: <index>
Byte3: <A LSB>
Byte4: <A MSB>
Byte5: <B LSB>
Byte6: <B MSB>
Byte7: <C LSB>

ID: 0x100
Byte0: <C MSB>
Byte1: '\n'

If commands have no return values then it is possible to send multiple commands in one go either inside a single CAN packet or split across multiple packets.

For example here are commands to control the state of the relay and the gripper position packaged up into a single CAN packet.

ID: 0x100
Byte0: ':'
Byte1: 'E'
Byte2: <value>
Byte3: '\n'
Byte4: ':'
Byte5: 'N'
Byte6: <value>
Byte7: '\n'

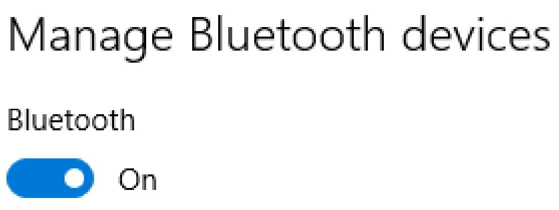
Bluetooth set up on Windows

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 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 and wait for the LEDs to stop flashing. If you don't know the Bluetooth name or Pair key then the defaults are MatrixArm and 1234. These defaults can be edited using the configuration tool and a USB connection to the Arm.

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 V2 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. If multiple Arms exist within the same proximity then it is a good idea to rename the Arms using the configuration tool and then label the Arms so you know which is which.

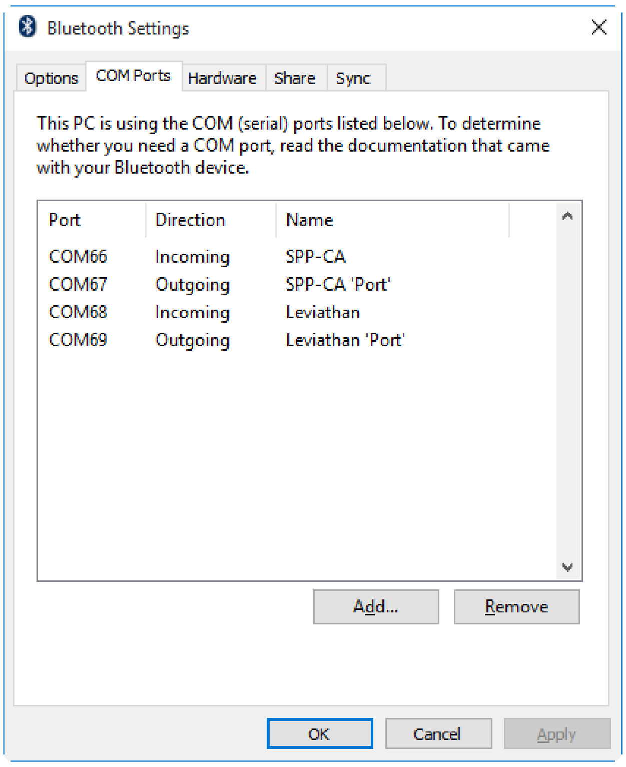
3) Determine the COM port number
You should get a popup balloon on the task bar saying device is ready to use. If you click this before it fades away then you can find out the COM port assigned to the robot.

You use this COM port number when communicating with the AllCode Robot Arm V2 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.

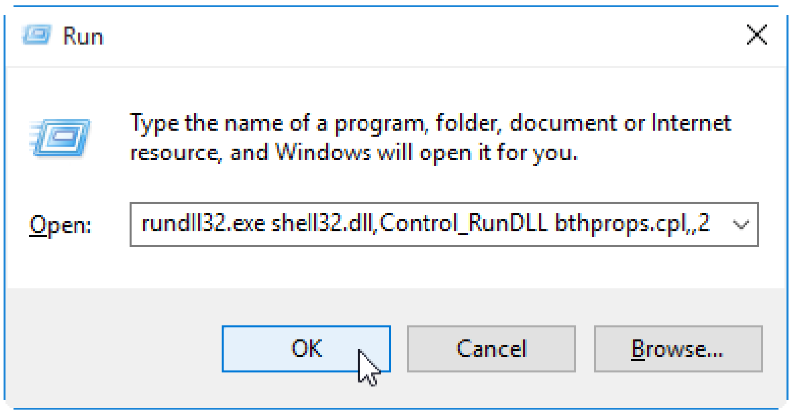


This PC is using the COM (serial) ports listed below. To determine whether you need a COM port, read the documentation that came with your Bluetooth device.

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

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 V2.

Bluetooth set up for Android

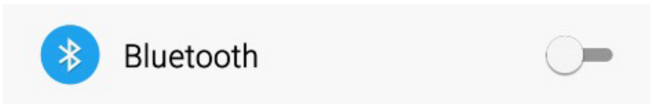
Android phones and tablets, when used with intuitive programming software such as App Inventor, provide a motivating platform for controlling the AllCode Robot Arm.

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

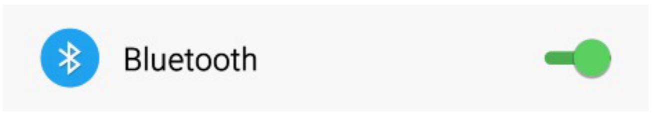
As with other devices, the AllCode Robot Arm must be paired with the phone or tablet before it can be used.

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



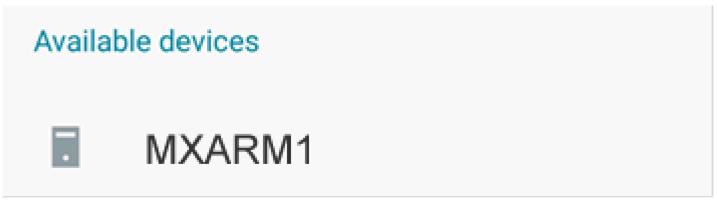
Bluetooth Switched On



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

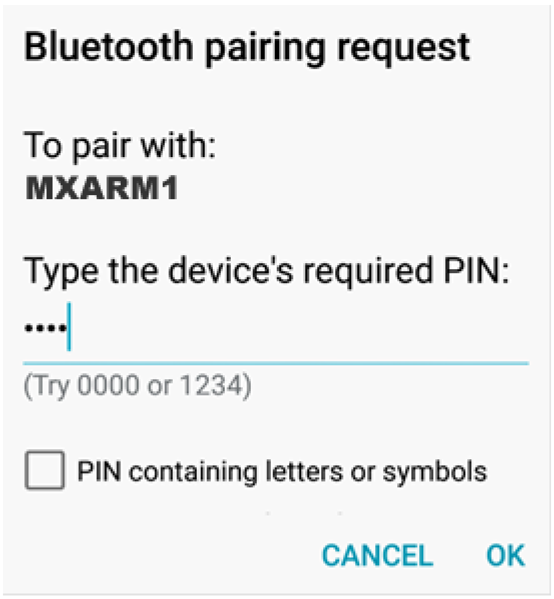
MXARM1

The name of the AllCode Robot Arm defaults to MatrixArm but can be changed via the configuration tool and a USB connection to a PC.

Bluetooth set up for Android

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

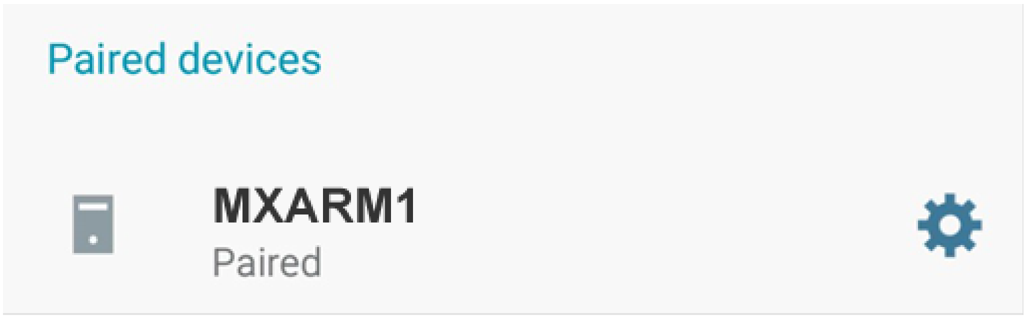
To pair with:
MXARM1

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.

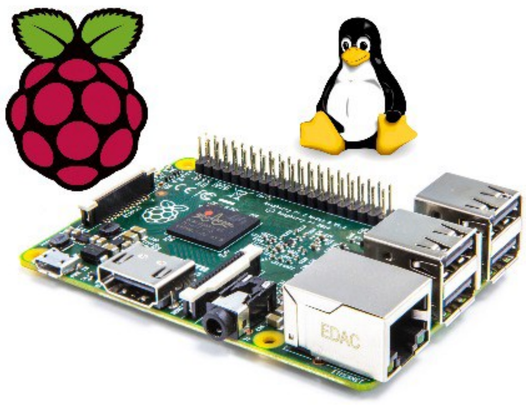


Paired devices

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.

Programming with RPi and Linux



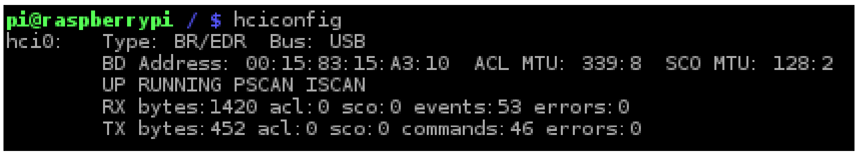
The Raspberry Pi is a popular single-board computer.

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.

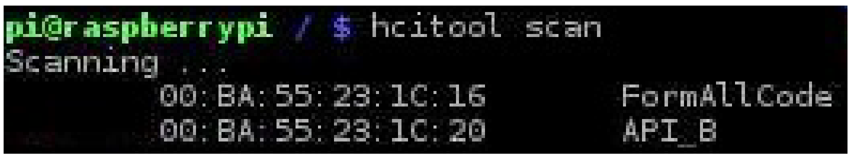
Setting up Bluetooth is relatively easy on a Raspberry Pi and can be done in a number of ways. The following steps are perhaps a more complex way of setting it up, but it should work in all situations. Note the Pi needs a Bluetooth USB dongle.

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":



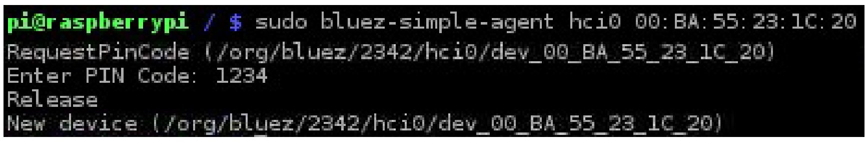
```
pi@raspberrypi / $ hciconfig
hci0:   Type: BR/EDR  Bus: USB
        BD Address: 00:15:83:15:A3:10  ACL MTU: 339:8  SCO MTU: 128:2
        UP RUNNING PSCAN ISCAN
        RX bytes:1420 acl:0 sco:0 events:53 errors:0
        TX bytes:452 acl:0 sco:0 commands:46 errors:0
```

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,



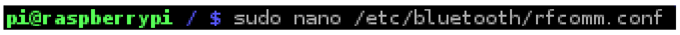
```
pi@raspberrypi / $ hcitool scan
Scanning ...
        00:BA:55:23:1C:16       FormAllCode
        00:BA:55:23:1C:20       API_B
```

"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 V2";
}
```

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 to output a write LED command:

```
echo ":a\x02\x01\n" > /dev/rfcomm1
```

If all goes well, LED A on the Robot Arm PCB should light up.

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 write LED command should now work ok.

Programming with Flowcode

For much of this course you can use the pre-prepared PC based apps or the Application Programming Interface commands.

However it is also possible to program the microcontroller on the circuit board directly and even make your own API.

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 Flowcode App developer, and;
2) re-programming the firmware on the robot.

For case 1 you can assume that the robot arm is ready to go. If someone else has been using the robot arm for low level microcontroller programming you may need to reprogram the API into the robot arm controller. Instructions for this are given in the Reference section: 'reinstalling the API'.

For case 2 it is possible to develop a microcontroller program in Flowcode or C and download it into the arm using the internal bootloader. Instructions for this are given in the Reference section: 'Reinstalling the API.
Note: Downloading code to the dsPIC E-block will remove the in-built API functionality from the robot and you will need to restore the API in the microcontroller. Again refer to the instructions in the Reference section: 'reinstalling the API'.

In Flowcode 9 and later there is a component available from the mechatronics menu to allow you to easily control the Robot Arm via the command API. This SCADA-like mode of operation allows you to program the robot arm using Flowcharts, Pseudocode, or blocks whilst the arm is connected to the PC using Wi-fi, Bluetooth or USB. Information on the Flowcode simulation component to allow you to do this is given in the Flowcode Wiki.

Programming with App Inventor

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.

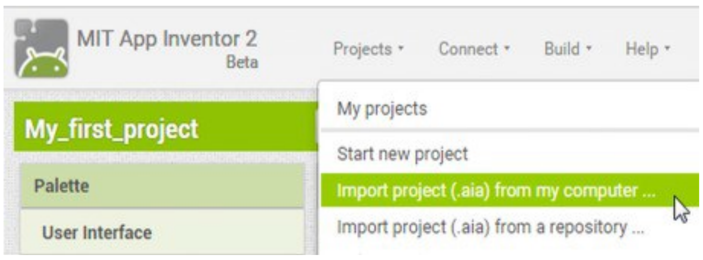
These QR codes and hyperlinks will help speed-up your installation, so you can start having some fun coding.

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.

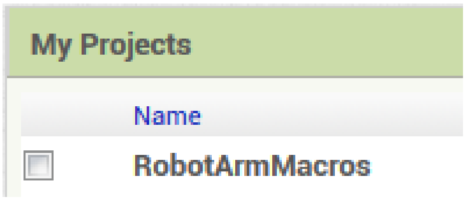
1. Set up a Google account (if you haven't already got one).
2. Go to the App Inventor website by scanning the QR code or clicking the hyperlink above and then login using your Google account.
3. Follow the online instructions, including installing the "MIT AI2 Companion App" onto your Android device.
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.

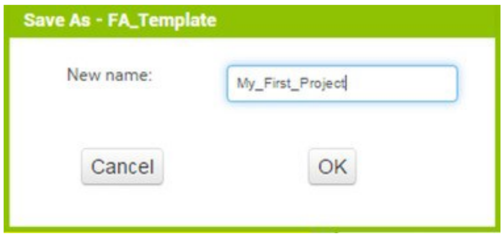


Your first program

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 ‘RobotArmV2Macros’ project.

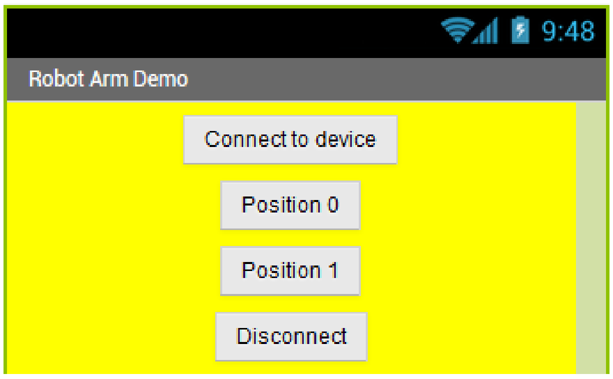




2. Save this template as a new file by selecting ‘Save project as…’ from the ‘Projects’ menu and then entering an appropriate name for your project.

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

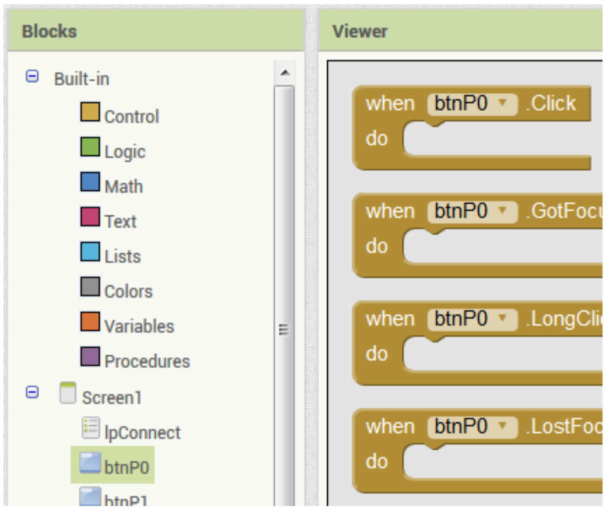
4. Drag a button from the User Interface panel onto the Viewer screen and alter its text to read “Position 0”. Also rename the button so it reads “btnP0”. Do the same again to create another button called “Position 1” with the name “btnP1”.



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

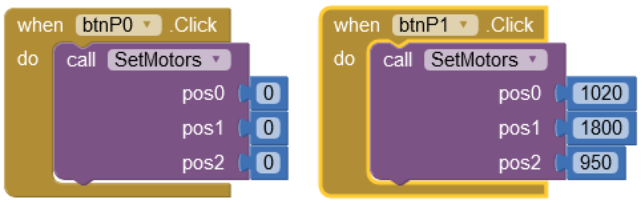
Drag the “when btnP0.Click” icon onto your program.

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



6. Click ‘Procedures’ from the ‘Built-in’ list and drag the “call SetMotors” 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 SetMotors code blocks.

Your two blocks should look something like this:



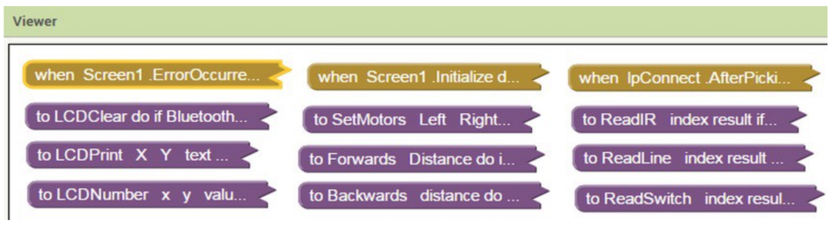
```
when btnP0 .Click
do  call SetMotors
            pos0  0
            pos1  0
            pos2  0

when btnP1 .Click
do  call SetMotors
            pos0  1020
            pos1  1800
            pos2  950
```

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 MatrixArm from the list. Clicking the “Position 0” button should make the robot go to it’s home position of 0,0,0, and “Position 1” should make the arm go to a 45 degree position with the arm reaching out down towards the table.

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.



```
Viewer
when Screen1 .ErrorOccurre...    when Screen1 .Initialize d...     when IpConnect .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 arm.

Programming with C++ / C# / VB

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/

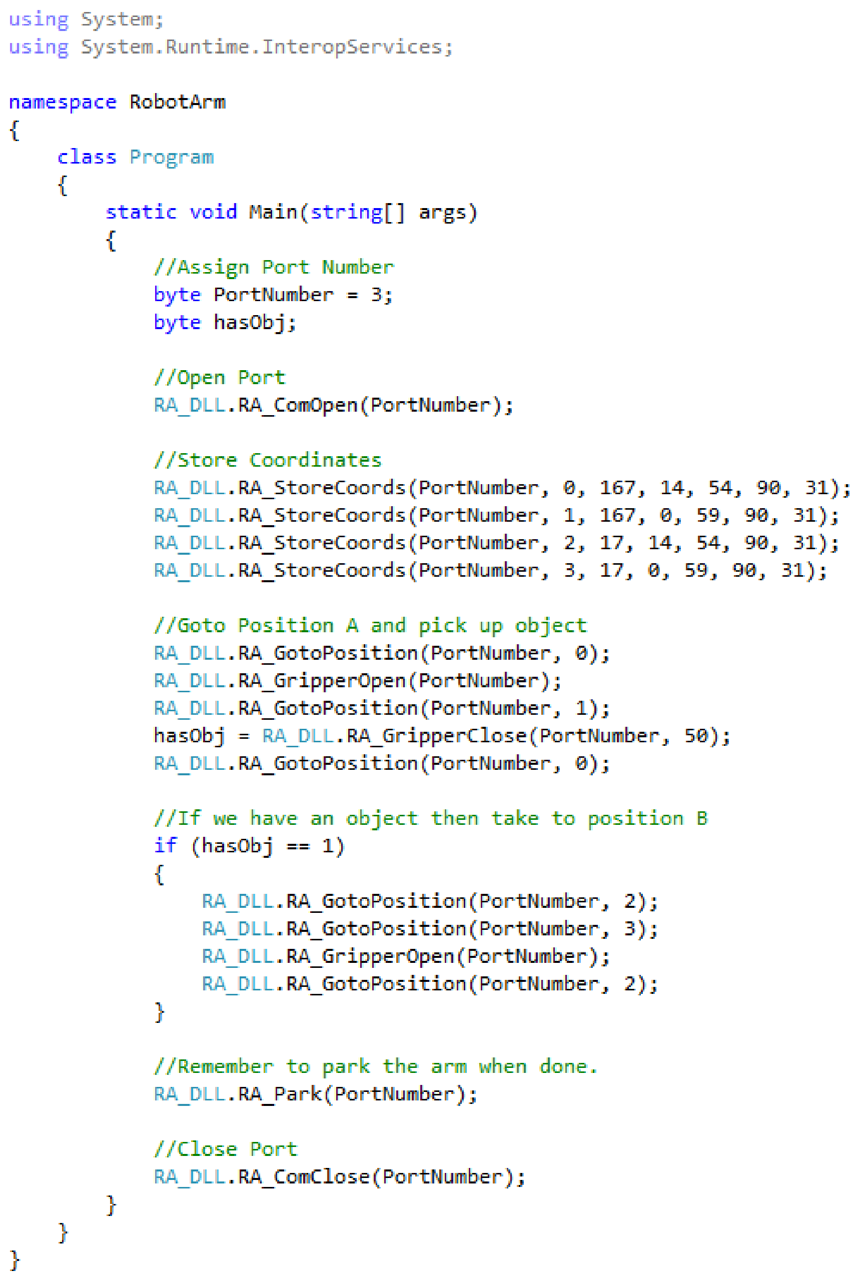
Using C#
The program on the right shows a basic program in C#.

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);
        }
    }
}
```

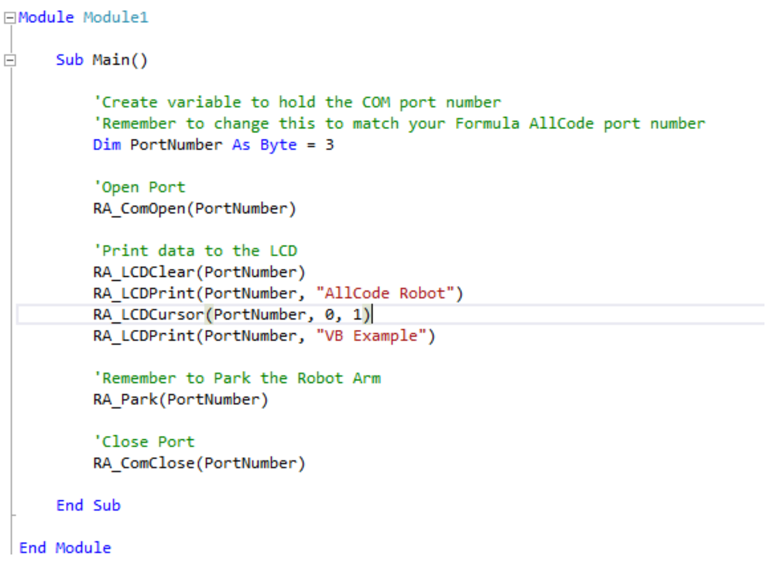
Programming with C++ / C# / VB

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.



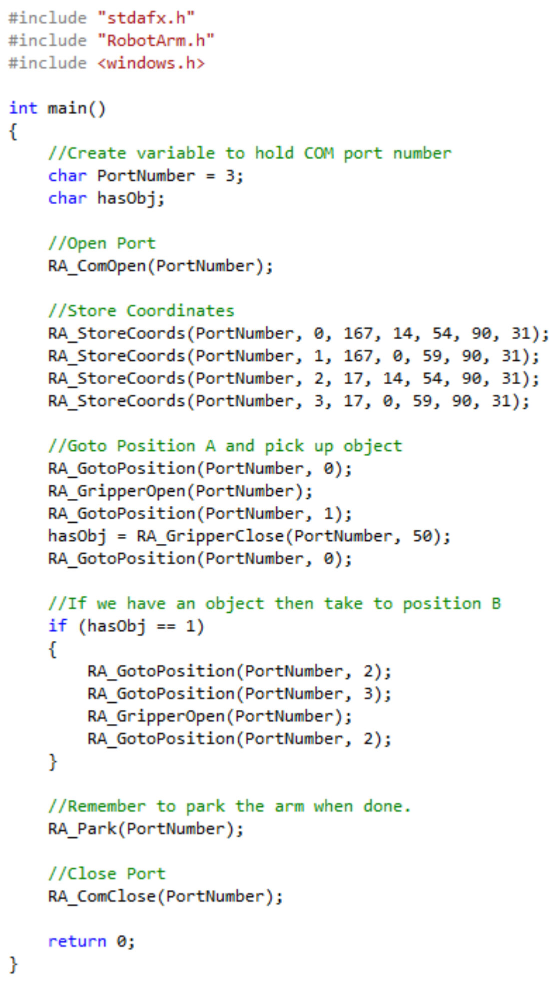
```
Module Module1

    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;
}
```

Programming with 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.

Programming with LabView

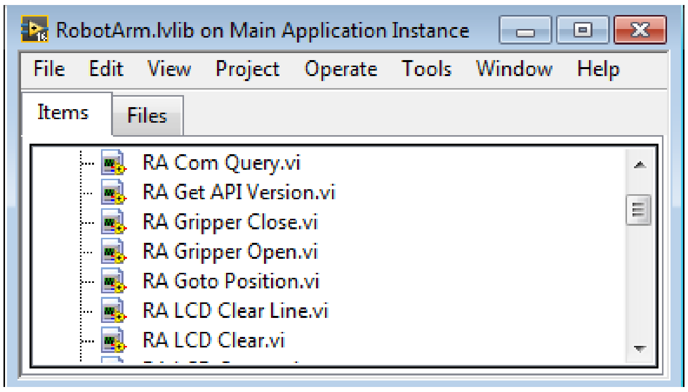
LabVIEW is a development environment for creating custom applications that interact with real-world data or signals in fields such as science and engineering.

It can also be used to control the AllCode Robot Arm.

This section explains how to get started

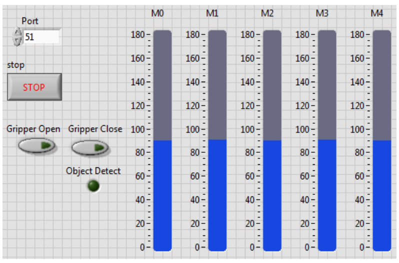
Using the robot with Labview is fairly easy and consists of using a library provided by MatrixTSL. First, download the library (which consists of a DLL and a LabView library file) from the Matrix TSL website:

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


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

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.

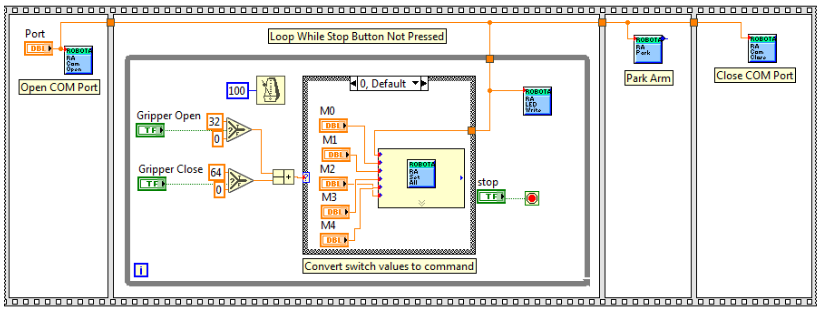


The program at the bottom of the page shows a flat sequence structure that ensures the various parts of the program are called in turn.

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.



Programming with PLCs and CAN bus

Programming the robot arm from a PLC largely depends on which type of comms interface you are using. There are likely two basic methods that will be the most simple to implement.

1. Over the network connection using either Ethernet or WIFI
2. Via CAN bus if available on your PLC

Network Communications

To communicate using a network first ensure that the arm is already configured and connected to your network using the information on the next page. You will need the IP address and port of the arm.

Then follow these steps to communicate with the arm.

A. Initialise the networking hardware
B. Open a TCP socket
C. Connect the TCP socket using the robot arm IP address and Port
D. Send the command bytes including command start, command code, parameters and command end bytes.
E. Attempt to receive any incoming return bytes from the arm depending on the command issued
F. Close the socket
G. Repeat steps A to F for each command you want to send to the arm

If communicating via the internet then you will need the external IP address of the network local to the arm. That network needs to be setup so that the port for the arm is forwarded to the local IP address for the arm.

CAN bus communications

For details on the CAN bus message format please refer to page 32.

Other Methods

Other methods to talk to the robot arm include Bluetooth communications if available on your PLC. To communicate via Bluetooth you will need to pair with the robot arm and then once paired you can connect and stream byte data in and out of the arm.

Bluetooth setup on the arm

The WIFI and Bluetooth communications are setup using the Configure software shown on page 24.

Configuring Bluetooth

To configure the Bluetooth connection first connect to the robot arm using a USB cable.

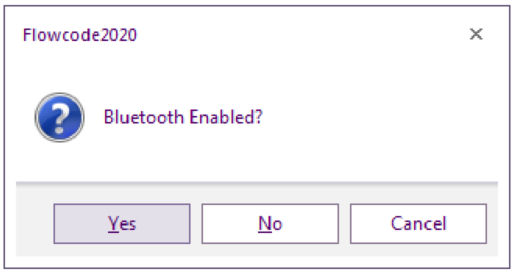
1. With communications set to USB click the Play button at the top left of the screen to start the program running.
2. The communications icon should turn to a Green Tick to indicate comms to the arm have been established.
3. Click the Configure Bluetooth button. You will be asked to enable the Bluetooth device, then for a name for the arm you are connected to, then for a PIN.
4. Once you have configured the Bluetooth connection you will need to switch the robot arm off and back on to allow it to setup the Bluetooth module as required.

Any details that you have setup using the configure software will be retained in none volatile memory so you should only have to re-run the configure software if you want to edit the settings.

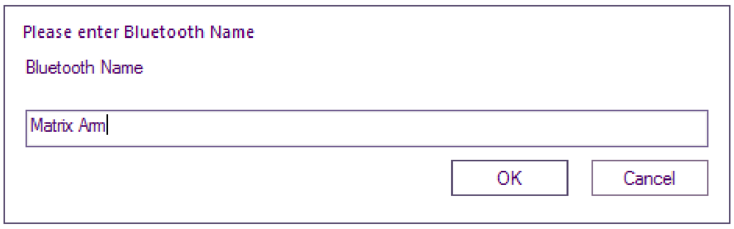
You will be able to run a number of arms in the same room.

Once you have done this you will need to pair the arm with your computer and set the COM port.

To operate any of the Apps using Bluetooth you will need to set the Comms to Bluetooth and set the Com port.

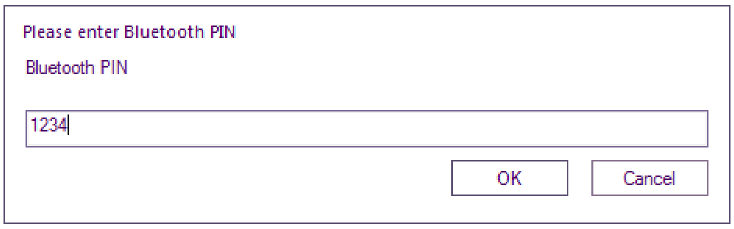

Flowcode2020

Bluetooth Enabled?


Please enter Bluetooth Name

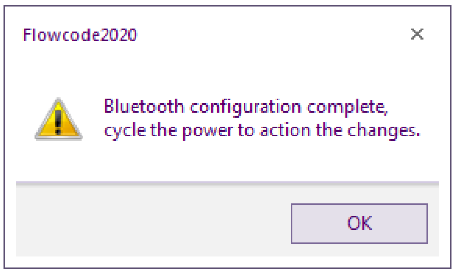
Bluetooth Name

Matrix Arm


Please enter Bluetooth PIN

Bluetooth PIN

1234


Flowcode2020

Bluetooth configuration complete, cycle the power to action the changes.

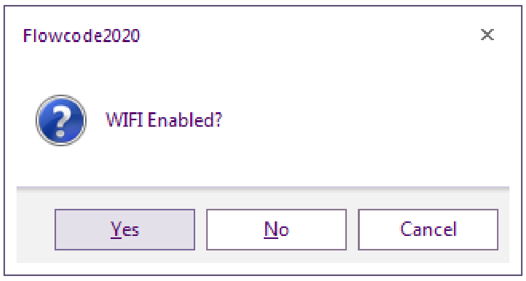
Wi-Fi setup on the arm

To configure the WIFI connection first connect to the robot arm using a USB cable.

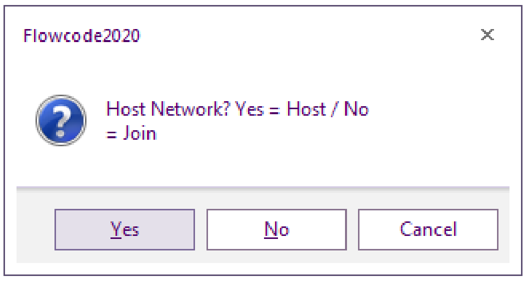
1. Click the Play button at the top left of the screen to start the program running.
2. The communications icon should turn to a Green Tick to indicate comms to the arm have been established.
3. Click on the Configure WiFi button. You will be taken through a number of dialogues as shows on the right.
4. Click the stop button at the top left of the screen.
5. Once you have configured the WIFI connection you will need to switch the robot arm off and back on to allow it to setup the WIFI module as required.
6. Once the robot arm has been switched back on click the Play button at the top left of the screen
7. Click the WIFI Get IP button to collect the IP address assigned to the robot arm by the network

Any details that you have setup using the configure software will be retained in none volatile storage so you should only have to re-run the configure software if you want to edit the settings.

If you want the robot arm to be connected from the internet then you need to enable port forwarding on your router to forward any external requests on the specified port onto the local IP address of the arm. You must then share the external IP address of the router as well as the port.

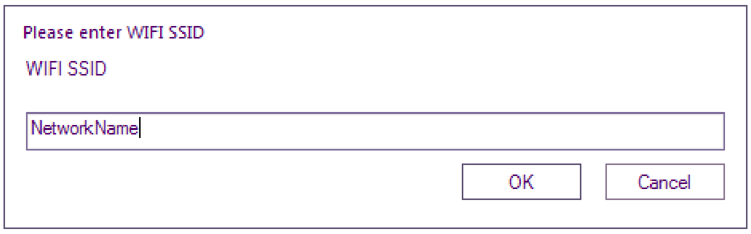

Flowcode2020

WIFI Enabled?


Flowcode2020

Host Network? Yes = Host / No = Join

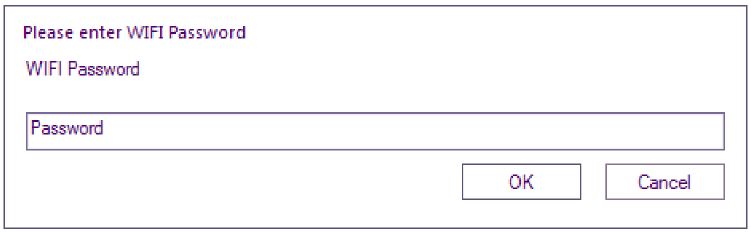
Select NO


Please enter WIFI SSID

WIFI SSID

NetworkName

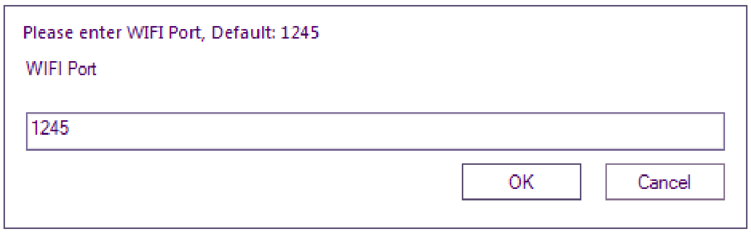
Enter the Wireless network same


Please enter WIFI Password

WIFI Password

Password

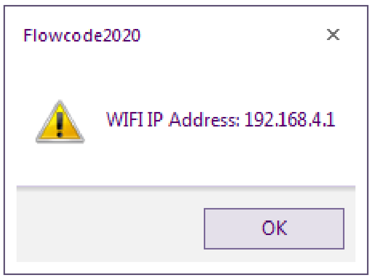
Enter the Wi-Fi password


Please enter WIFI Port, Default: 1245

WIFI Port

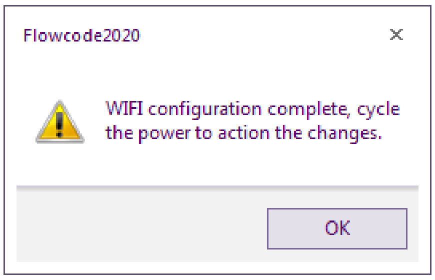
1245

Use default 1245


Flowcode2020

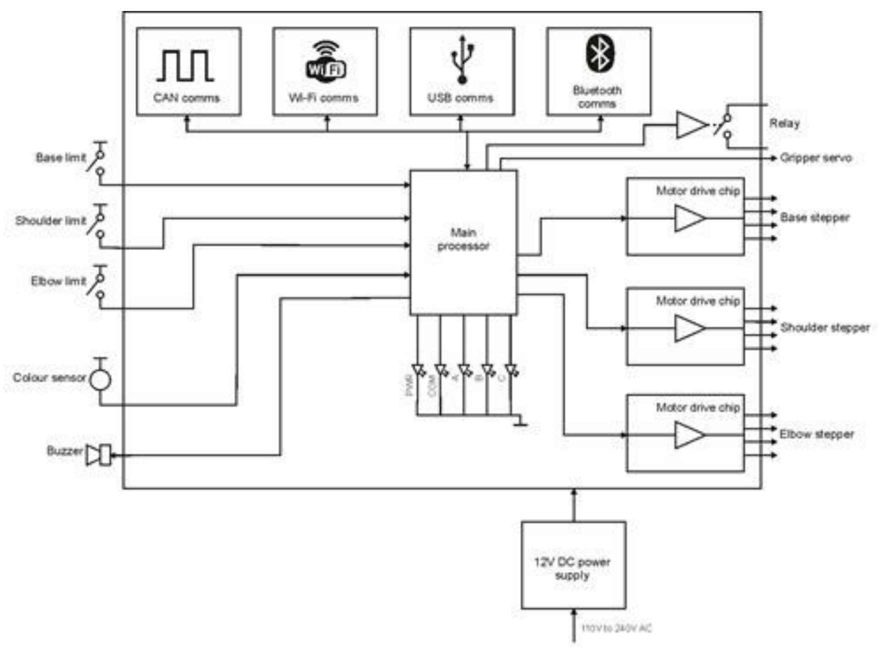
WIFI IP Address: 192.168.4.1

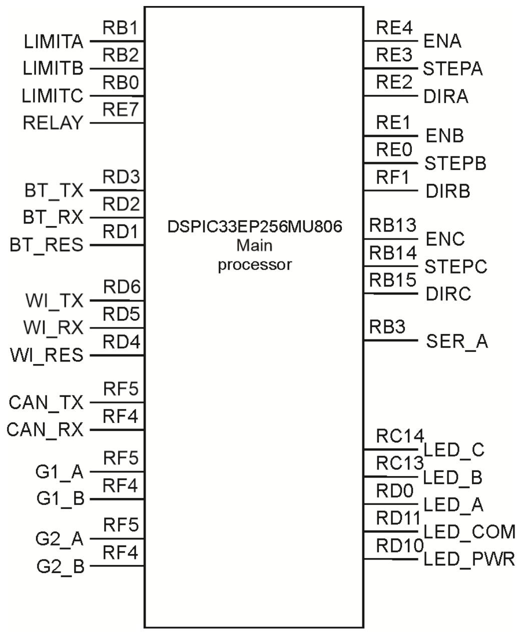
This is the address set by your router
You will need to enter this into the App.


Flowcode2020

WIFI configuration complete, cycle the power to action the changes.

Connections and circuit





If you want to reprogram the microcontroller at a low level then you will need to understand which pins of the microcontroller connect to the other parts of the robot arm. You can get this data from the diagram above.

The microcontroller used is a DSPIC33EP256MU806 dsPIC.

The motor driver outputs feed A3967 driver chips. These convert Enable, Step, and Direction into pulses for stepper motors. These chips make use of 8:1 microstepping technology to provide steps of 0.044 degrees per step.

The Bluetooth module is a RN4678. The Wi-fi module is a ESP12F. The CAN bus driver is a MCP2551.

The gripper is controlled by a small servo motor. The output on SER_A has as 20mS pulse for this-purpose. The RELAY output is buffered by a transistor and it activates the on board relay with RELAY dictating whether the output from the relay is +12V or OV. This is for expansion with a vacuum sucker. The colour sensor, G1_A and G1_B, and buzzer, G2_A and G2_B, are Grove devices.

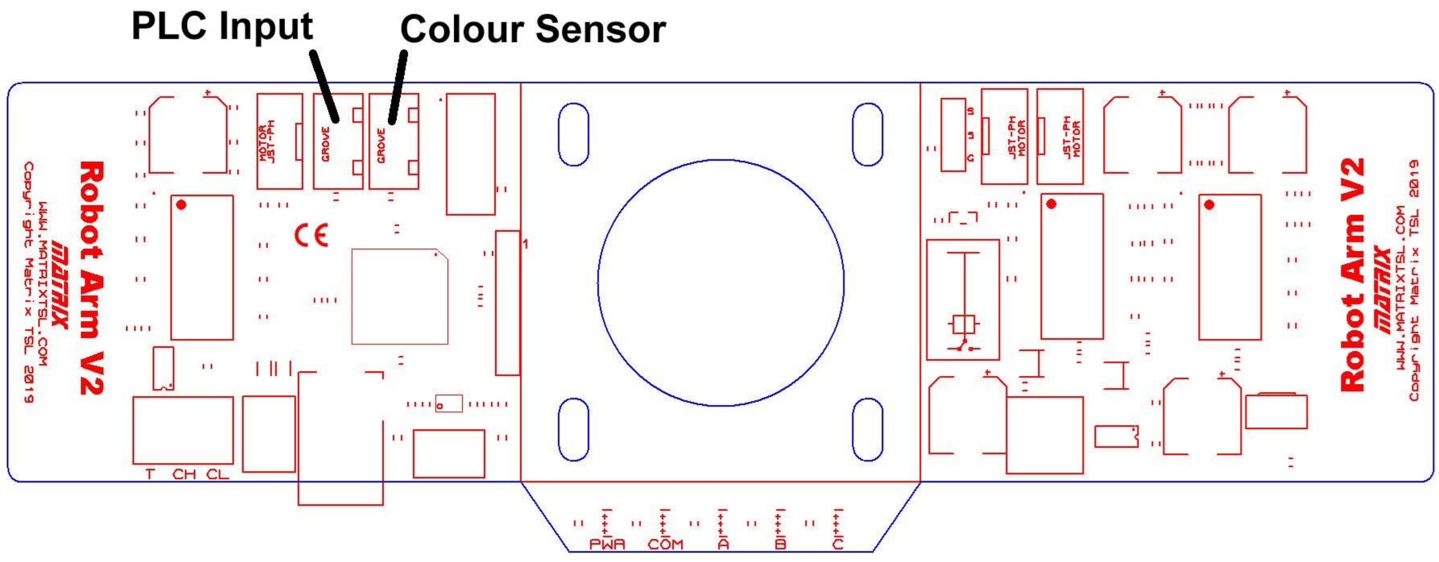
The CAN screw terminals are: CH (high), CL (low) and T for termination. T has a 120 ohm resistor to ground and can be used for termination if needed

Power connector is 2.1mm power jack centre positive.

Datasheets on all these items are available on the internet.

To understand how to transfer your program to the microcontroller please read the section on Reinstalling the API in the Reference section. This shows you how to send a hex program to the controller.

Connections and circuit

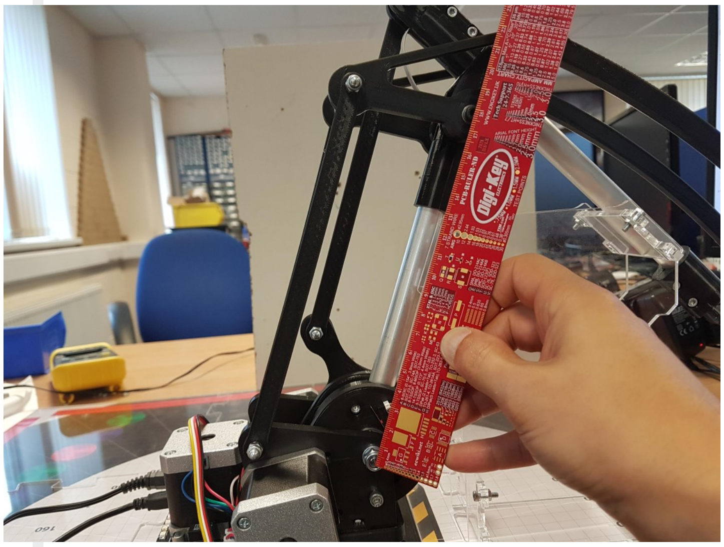


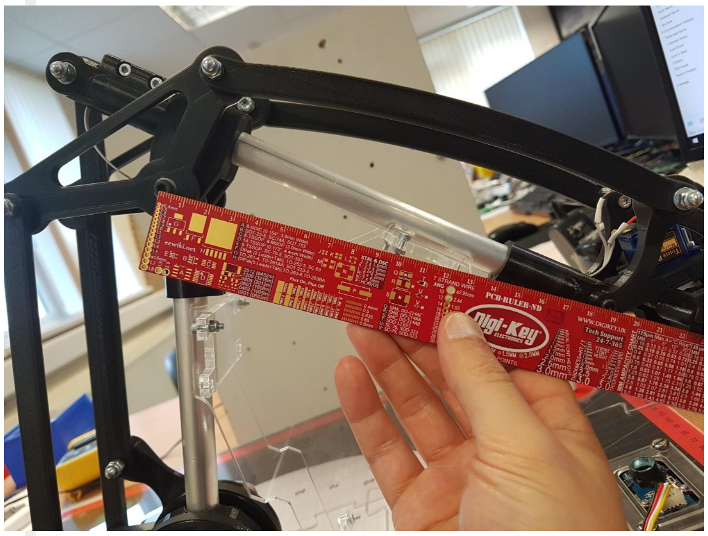
There are several steps in the calibration:

Entering Calibration
Run the Open Configure program.
Click the Go button and the Communications light should turn green
Click the Set Kinematics Lengths button and enter the lengths you recorded during calibration
Click the Motor Enable and Home button and the arm should go to the 90 degree position on the mat at around coordinates 0, 300, 5.

Calibrating Base Angle
If the centre of the gripper is off the 90 line then work out the angle difference.
Click the Set Motor Base Angle button and enter the angle difference. A little goes a long way try increments of 0.2.
The Arm should move to show the new position.
Once you have centered the arm on the line use the Motor Enable and Home button to sanity check that the value is stored and the setting is ok.
(Note from the initial start angle you cannot go back more then **-2.0** degrees, you may need to back off the base motor limit switch if you cannot go back far enough.)

Calibrating Shoulder and Elbow
Now we have the gripper centered over the X0 axis line it's time to calibrate the shoulder and elbow.
Click the Y-/Y+ Z-/Z+ buttons until the center of the gripper is over the Y300 line and 5mm from the surface of the jigsaw plastic mat. A 5mm spacer can be used to help get this accurate.
Once the gripper is in position click the Store Calibration button to calculate and store the calibration.
Click the Position 2 button
Using a ruler measure the offset from the central line, left of the line is positive, right of the line is negative.
Click the Set Drift button and enter the offset in mm.
Click the Position 2 button again to check we are now on the central line.
Close the Configure application


Measure and record the length of the shoulder, centre of main pivot to centre of bolt. **The default is 176.0**


Measure and record the length of the elbow, centre of bolt to centre of bolt. **The default is 189.0**

Support and fault finding

1) If you have any faults then it will help us and yourselves to understand a little more about the nature of the fault.

2) An Any App you can select VIEW….CONSOLE. This brings up the Flow-code console for the arm. Select the Robot Arm tab. This shows the messages between the API and the PC and effectively gives you a status of the commands being executed.

Communications between the arm and the Wi-fi or Bluetooth system sometimes presents issues: the API has various pauses in the code to allow for wireless communications.

On power up the arm looks for communications systems. In this phase of power up the LEDs take on a different meaning:

PWR: Power

COM: when flashing this shows the arm is in Bootloader mode.

A: Most significant bit

B: Middle bit

C: Least significant bit

Where flashing:

1 = USB

2 = CAN

3 = Looking for Bluetooth connection

4 = Wi-Fi

Version control

21 07 20	First start
15 02 21	small change p21
28 07 21	Changes concerning new G code IN1 and IN2 functions
05 10 21	Calibration information updated
10 08 23	Reformatted to new style