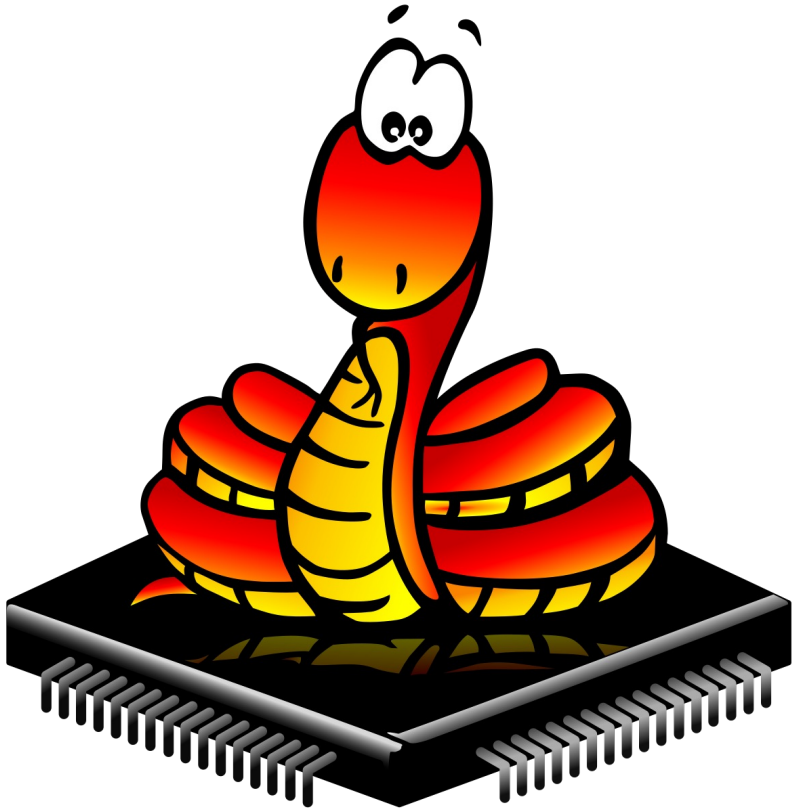
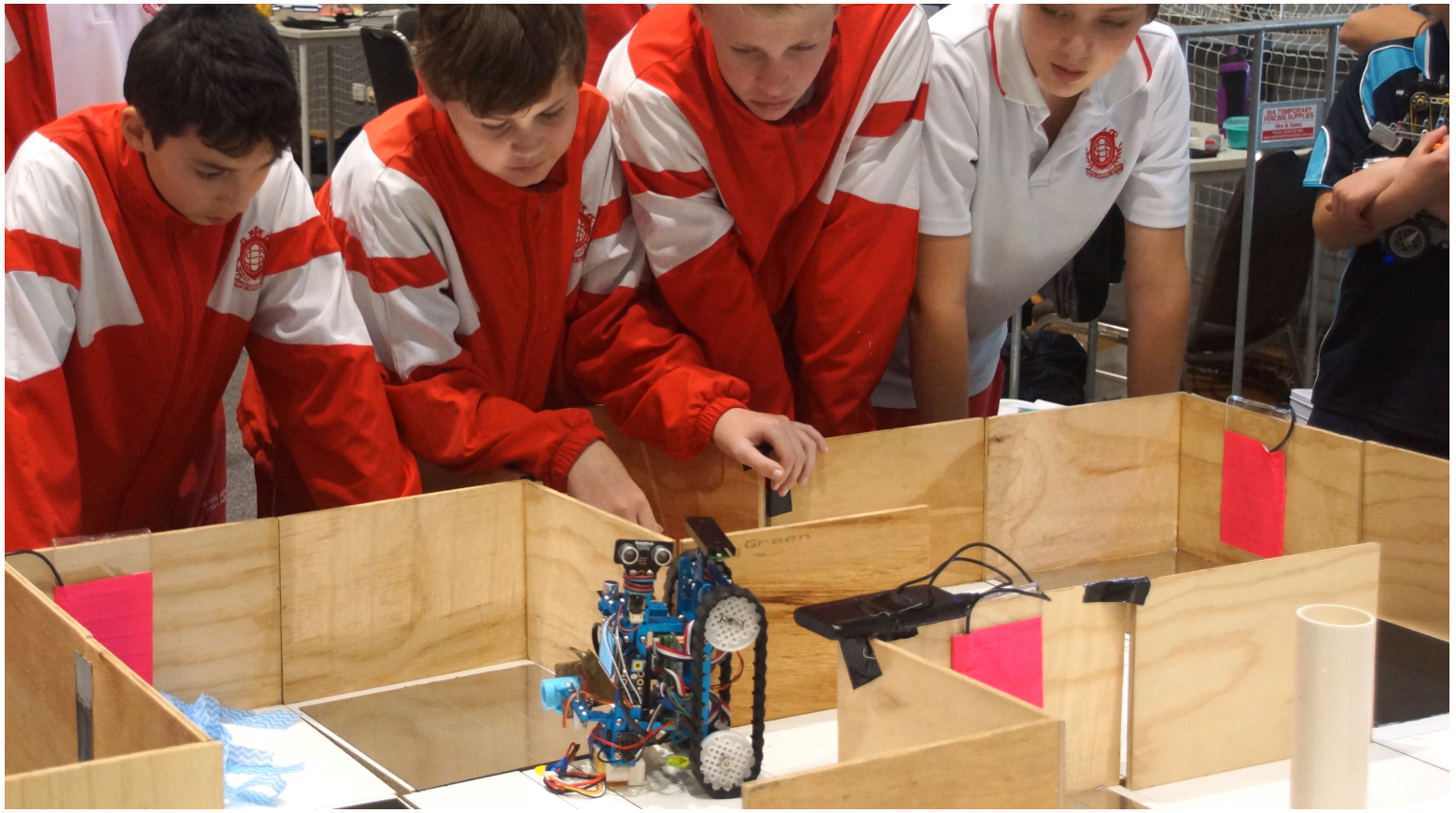




RoboCup Junior Australia

MAZE RESCUE FOR EV3 USING MICROPYTHON

RoboCup Junior AUSTRALIA

Overview of Competition

- "Real world" robot challenge
- Modular, flexible game design
- Accessible to a wide range of robot platforms
- Many possible solutions
- Easy for beginners but with a high ceiling
- Simple scoring
- Engaging and fun.
- Compatible with RCJ International Rules

Now even easier for beginners!

CAN BE COMPETITIVE WITH AN EV3

CAN USE THE EV3 CLASSROOM SCATCH PROGRAMMING

PATHWAY TO MORE ADVANCED SOLUTIONS

LOTS OF DESIGN FLEXIBILITY

Scenario

- There has been a disaster in a factory building.

- Several workers are trapped inside but it is not safe to send in rescue teams. They are wearing high-vis clothing and some have short-range transponders

- The role of your robot is to enter the building, identify victims, and leave them a package that will help sustain them until the rescue teams arrive. The robot should then travel back to the Exit for extraction

The Competition

You have 4 minutes to calibrate your robot, find the victims and exit the maze.

There is a minimum of 5 victims in the maze. You get points for each one you find

Bonus points for leaving rescue packages and for exiting the maze (if the majority of victims have been found) and for finding Checkpoints

Victims on “floating” walls score more points.

The robot must avoid black holes

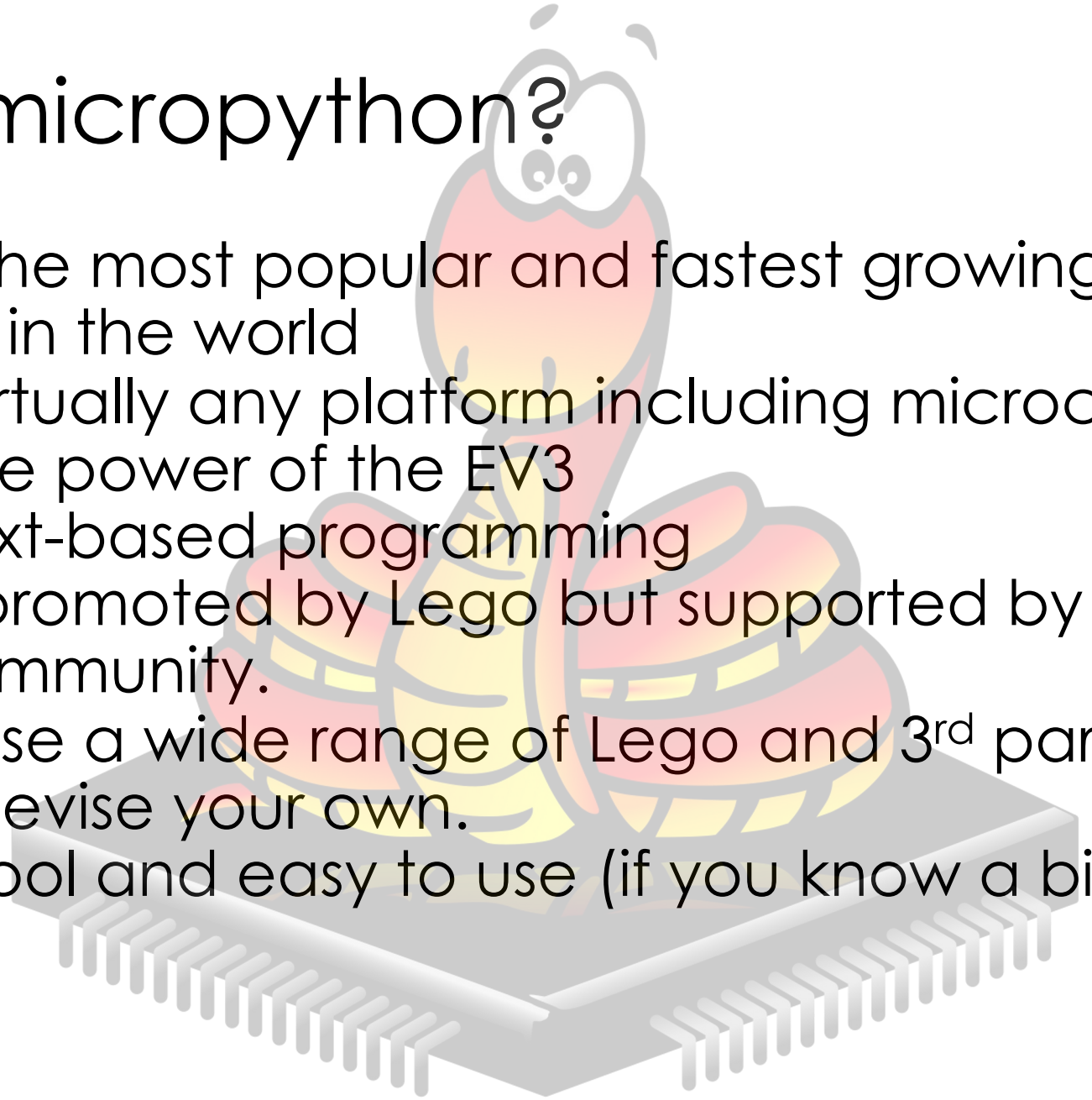
The robot can be return to last silver checkpoint if stuck without penalty but program cannot be restarted

Robot run can be restarted with loss of points at any time

READ THE RULES !!!!

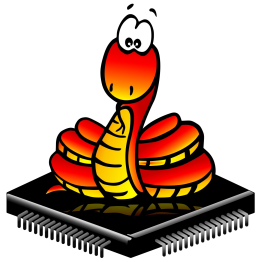
CLICK HERE TO SEE A SHORT VIDEO

Why use micropython?



- Python is the most popular and fastest growing programming language in the world
- Runs on virtually any platform including microcontrollers
- Unlocks the power of the EV3
- Flexible text-based programming
- Officially promoted by Lego but supported by ban open-source community.
- You can use a wide range of Lego and 3rd party sensors or you can devise your own.
- It is very cool and easy to use (if you know a bit about Python).

Getting started with micropython



For the bulk of this document we are assuming that you have a basic concept of the fundamentals of python programming. There is a lot of materials available online to help you get started.

The following links and videos will get you started. It is highly recommended that you can get a basic program up and running before proceeding through this document.

Here is the official Lego page …
https://education.lego.com/en-us/product-resources/mindstorms-ev3/teacher-resources/python-for-ev3

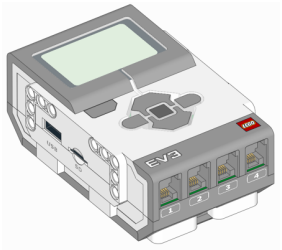
The online manual
https://pybricks.com/ev3-micropython/

Installing Visual Studio Code
https://www.loom.com/share/d102dfc9e9d14a04bbf3d8ab72101bf7

Setting Visual Studio Code up for EV3
https://www.loom.com/share/a2741e0f7a974dcaba6697c994bc2691

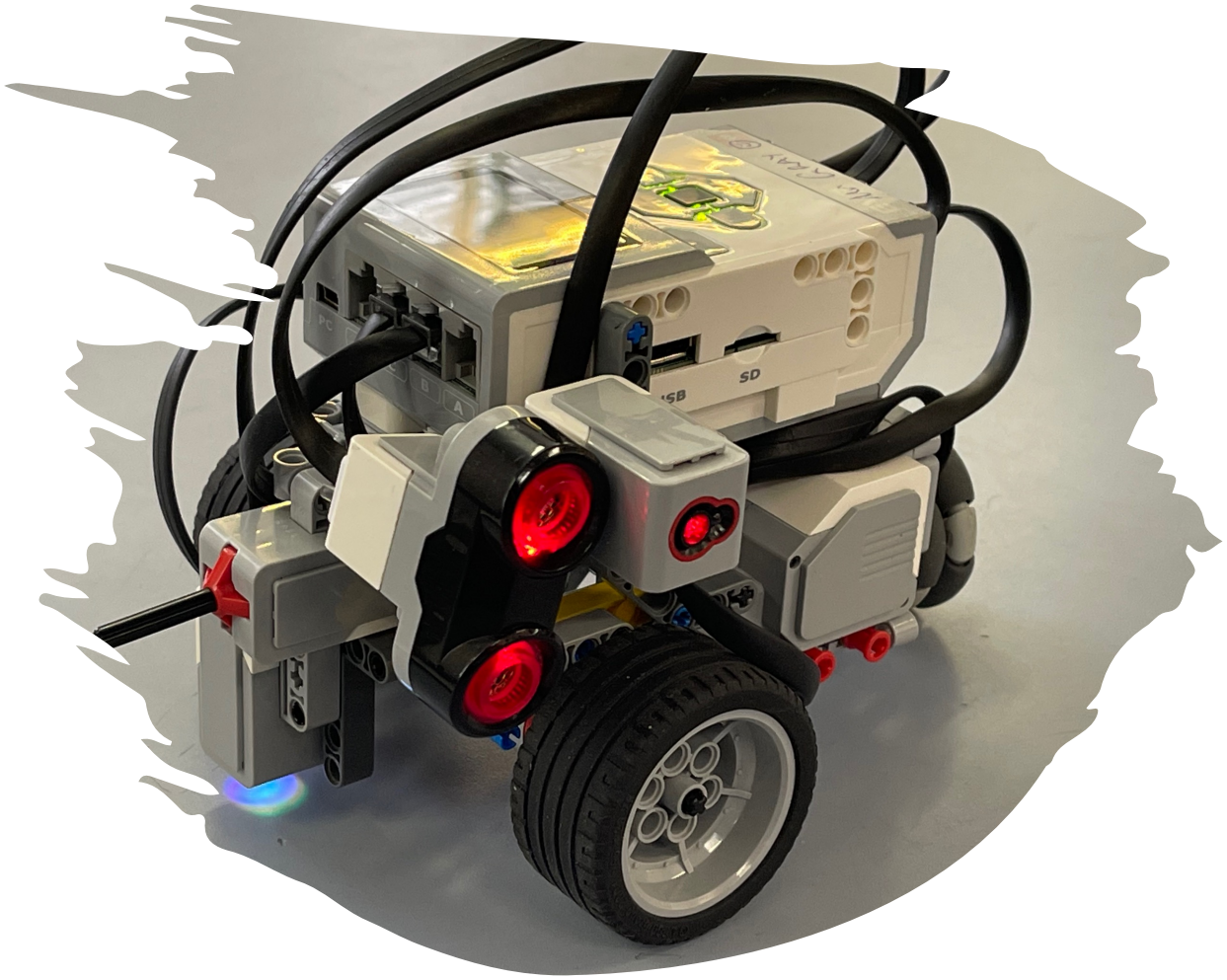
Your first program
https://www.loom.com/share/bff034775fe446efb344736d3a0a9db3

Here is an excellent playlist of basic EV3 micropython capabilities.
https://www.youtube.com/playlist?list=PLfZDz4HU7SLzQlsfXlfziaOhODz3eHCG



The Robot – what you need

- Basic EV3 robot
- Micropython SD card
- Colour sensor pointing down
- Option : Mindsensors IR temperature sensor pointing at wall (or second Colour sensor)
- 1 X Ultrasonic (facing same wall as IR temperature sensor)
- 1 X Touch sensor on front of robot (or second Ultrasonic)
- You will need to add a package dispenser and additional motor later
- It works !

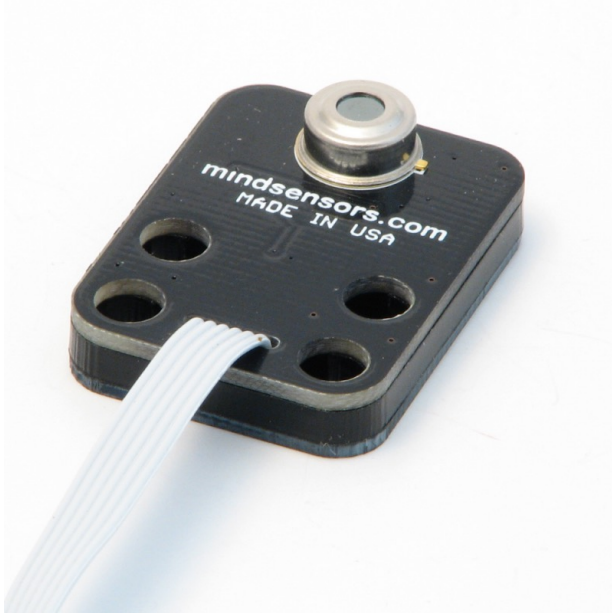


Measuring temperature

Mindsensors IR temperature sensor

http://www.mindsensors.com/pdfs/IRThermometer-User-Guide.pdf

- Non-contact temperature sensor
- Very responsive and accurate
- Can be used with (recommend) or without (as pictured) collimator.
- Measures ambient and target temperatures in °C and °F
- Raw results need to be modified in micropython before use (divided by 100)
- You will need to import the EV3devSensor library to your program.
- Available from MTA https://www.teaching.com.au/product/MIND001
- You can try the sensor out by connecting it to the EV3 and using Device > Browser >{port}> Watch Values

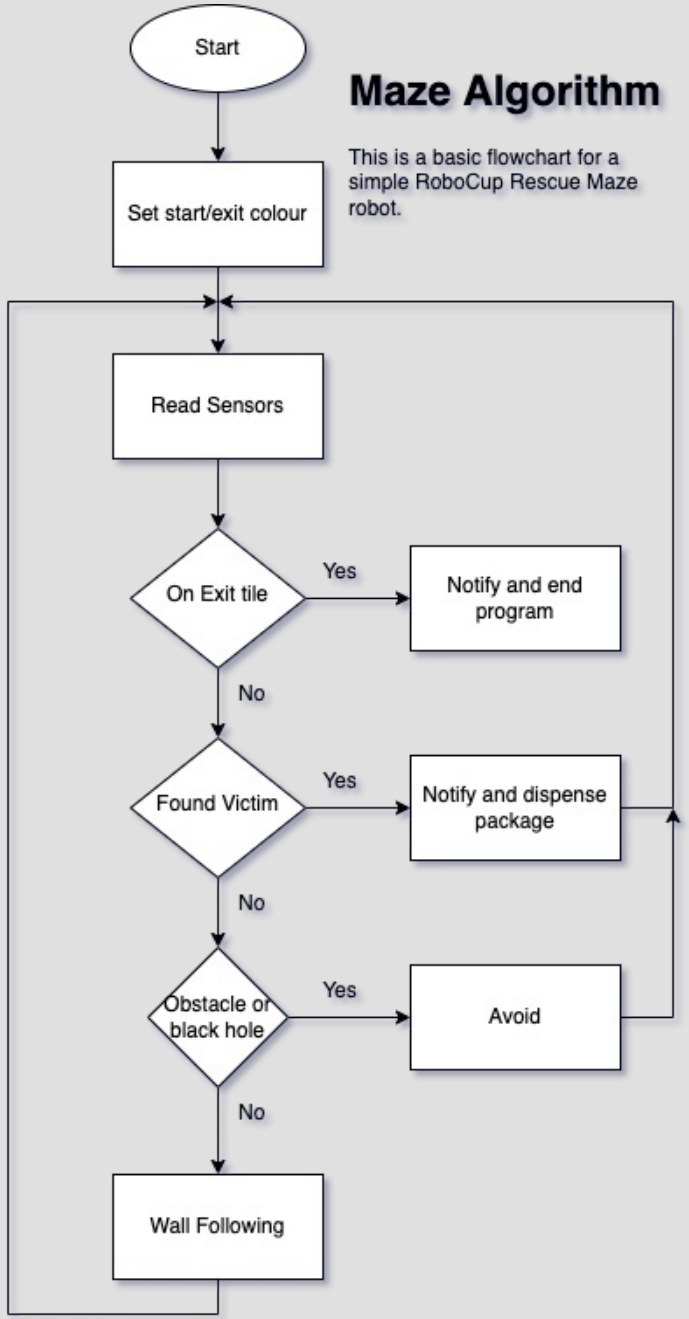


Maze Algorithm

- Keep it simple and work on one element at a time.
- Make your wall follower as smooth as possible. You want to stay at a constant distance from the wall.
- The rate at which the robot turns is important for reliable performance
- A good development strategy is

✓ Follow a wall
✓ Detect an obstacle
✓ Detect a black hole
✓ Find a victim
✓ Leave a package
✓ Record and identify entry squares
✓ Exit detection

We "nest" a series of if … else statements <u>one at a time</u> to get the robot behavior that we are after.



Basic Program setup

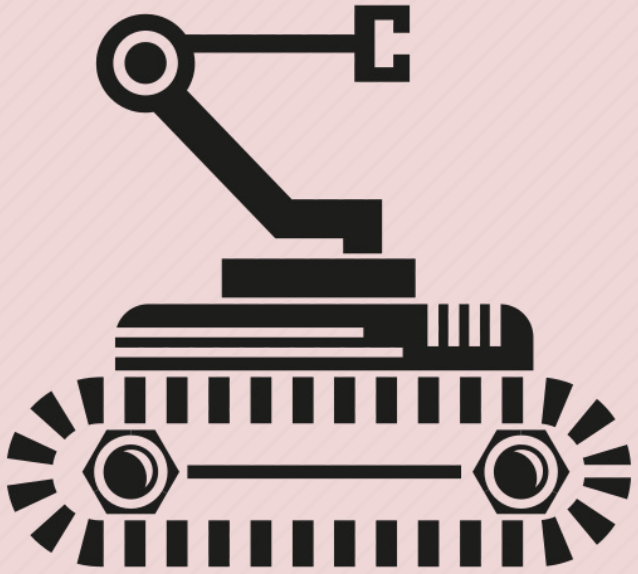
Open Visual Studio Code

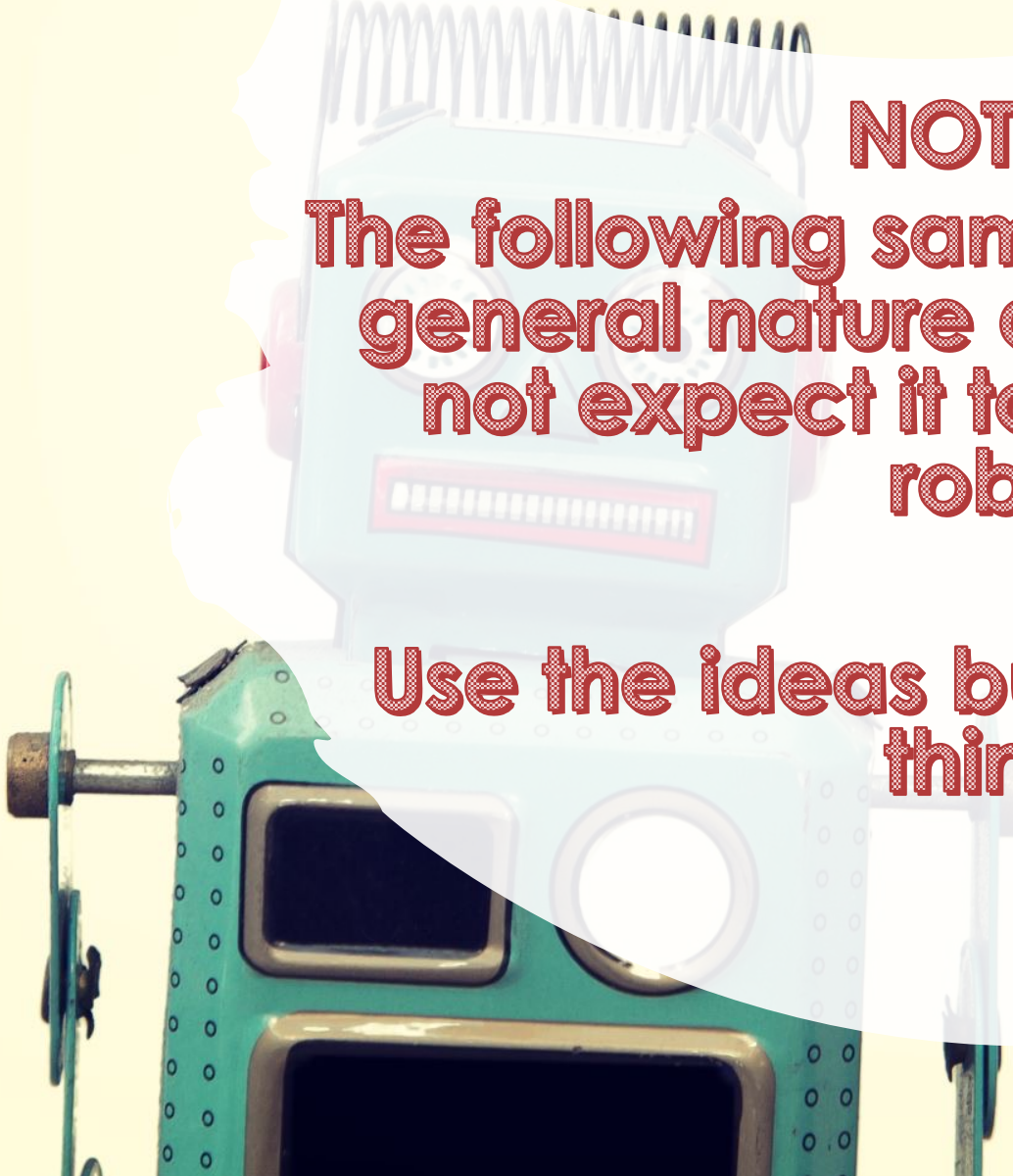
Make a new file using EV3

Add EV3dev library

Setup sensors and motors

Code away !





NOTICE

The following sample code is of a general nature and you should not expect it to work in your robot

Use the ideas but do your own thing!

Sample Robot Setup

- Light sensor pointing down
- Ultrasonic pointing at wall (locate forward of front wheels oriented vertically)
- Touch sensor pointing front (make sure that it operates reliably and doesn't jam)
- Mindsensors IR Temperature sensor pointing at wall (centre of victim)

Basic Setup

main.py

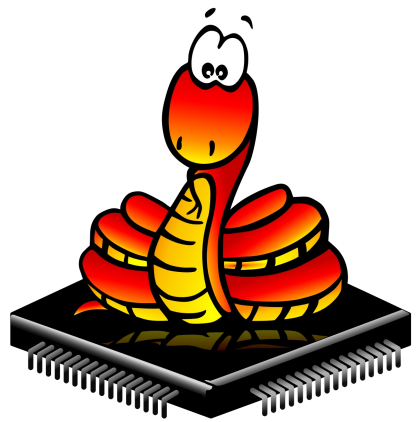
```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile

# Manually add this line for non Lego sensors.

from pybricks.iodevices import Ev3devSensor

# This program requires LEGO EV3 MicroPython v2.0 or higher.
# Click "Open user guide" on the EV3 extension tab for more information.


# Create your objects here.
ev3 = EV3Brick()

# Sensor setup

side = UltrasonicSensor(Port.S1) # distance sensor facing wall
front = TouchSensor(Port.S3) # touch sensor on front
sensor = Ev3devSensor(Port.S2) # Mindsensors IR temperature facing wall
light = ColorSensor(Port.S4) # Color sensor facing floor

# Motor setup.

right_motor = Motor(Port.D) # drive motor
left_motor = Motor(Port.A)  # drive motor

# Write your program here.
ev3.speaker.beep()

```

Most of this stuff added automatically

You will need to add the additional sensor support

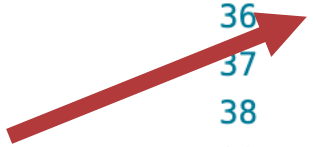
Define your sensors and name them

Setup the motors



Simple Wall Following

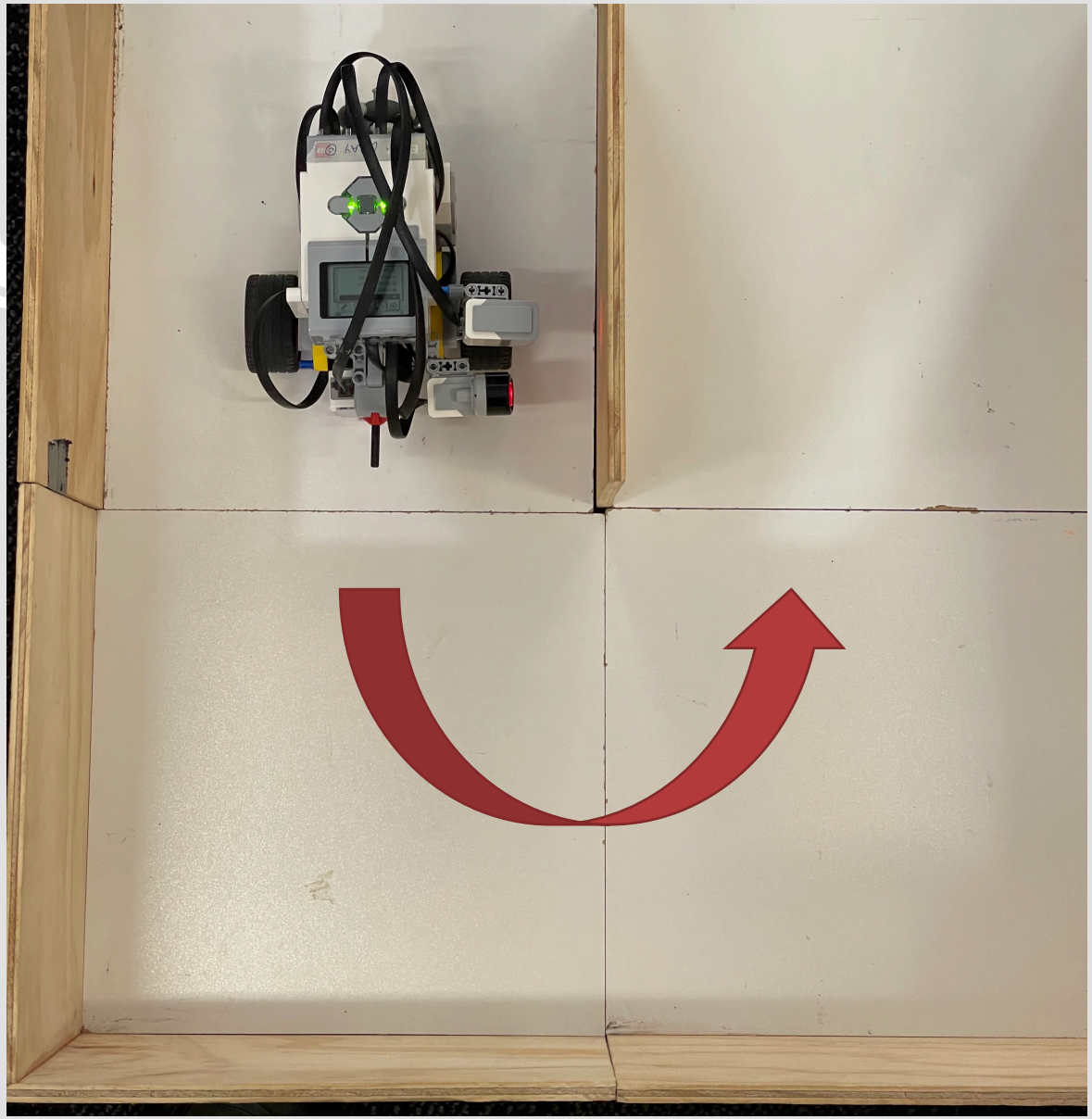
- Wall following is simple. The robot either steers towards the wall if too far away or away from the wall if too close.
- It is a good idea to set up some variables so you can easily tune the robot
- The key issue is carefully calibrating your robot turn rates. The robot should follow a constant arc around a wall divider.
- Individual motor speed control works better than steering



```
33  # Variables - makes tuning easier
34
35  base_speed = 50 # speed you want robot to run at
36  diff_speed = 25 # how much you will slow one of the motors for turns
37  setpoint = 120 # distance from wall in mm
38
39  # Write your program here.
40  ev3.speaker.beep()
41
42  while True: # looping forever
43      wall = side.distance() # reading value from ultrasonic (mm)
44      if wall < setpoint: # comparing to desired distance
45          left_motor.dc(base_speed) # too close - steer away
46          right_motor.dc(base_speed - diff_speed)
47      else:
48          left_motor.dc(base_speed - diff_speed) # too far away - steer towards
49          right_motor.dc(base_speed)
```

Setting the Turning radius

- When your robot comes to the end of a wall it will try to turn towards where it thinks the wall is.
- Setup your motor speeds so that your robot will do a smooth arc around the wall.
- There is a relationship between speed and turning plus every robot has different characteristics. Play around until you get it right.
- Individual motor speed works much better than using “steering”.



A better wall follower

- The problem with the previous sample is that the robot is always turning. Adding a tolerance zone where it just goes straight makes the robot much smoother and faster.
- We can calculate how far we are from the setpoint distance
- Using an elif (else .. if) means we can have multiple decision cases
- With these settings the robot will go straight for wall distances between 110 and 130mm.

```
33  # Variables - makes tuning easier
34
35  base_speed = 50 # speed you want robot to run at
36  diff_speed = 25 # how much you will slow one of the motors for turns
37  setpoint = 120 # distance from wall in mm
38  maxError = 10
39
40  # Write your program here.
41  ev3.speaker.beep()
42
43  while True: # looping forever
44      wall = side.distance() # reading value from ultrasonic (mm)
45      error = wall - setpoint
46      if error < -maxError: # too close
47          # steer away
48          left_motor.dc(base_speed)
49          right_motor.dc(base_speed - diff_speed)
50      elif error > maxError: # too far away
51          # steer towards
52          left_motor.dc(base_speed - diff_speed)
53          right_motor.dc(base_speed)
54      else: # just right !
55          # go straight
56          left_motor.dc(base_speed)
57          right_motor.dc(base_speed)
```

Proportional wall follower

- You may get even better results using proportional control.
- There is a relationship between the base_speed and the gain required. Too much gain and the robot will become unstable. Too little and the robot response will be too slow
- We calculate the error
- It is still important to limit the turn radius.
- If the robot is still not fast enough try using PID control

Video comparison
On/Off vs Proportional vs Fast

```
31
32  # Variables - makes tuning easier
33
34  base_speed = 50 # speed you want robot to run at
35  diff_speed = 25 # how much you will slow one of the motors for turns
36  setpoint = 120 # distance from wall in mm
37  maxError = 10 # maximum deviation from setpoint
38  gain = 1.2 # multiplier to set reaction speed
39
40  # Write your program here.
41  ev3.speaker.beep()
42
43  while True: # looping forever
44      wall = side.distance() # reading value from ultrasonic (mm)
45      error = wall - setpoint # calculating error
46      if error > maxError: error = maxError # limiting turn radius
47      if error < -maxError: error = -maxError
48      left_speed = base_speed - error*gain # calculating motor speeds
49      right_speed = base_speed + error*gain
50      left_motor.dc(left_speed)
51      right_motor.dc(right_speed)
```

Obstacle Avoidance

Here is a code snippet from inside a forever loop. This example just stops the robot.

- When you see an obstacle in front of the robot you need to react in some way.
- This can be a wall or an obstruction. Even a black hole can be treated as an obstacle as seen in the example on the right.
- If you see a wall in front of you for example you could reverse a little and then turn away from the wall.
- Keep the movements small – it is not unusual to take a few attempts at getting around the obstacle.

Hint: This is also putting messages up on the screen (more later)

```
if front.pressed() or light.color() == Color.BLACK: # found wall or black
    # do something if you run into the wall or find black
    ev3.screen.clear()
    ev3.screen.print("OOPS")
    left_motor.dc(0) # in all these cases the motors are just stopped
    right_motor.dc(0) # you need to do your own thing.
    wait(1000)
    ev3.speaker.beep()
else:
    # your slightly clever wall following code
```

Finding the Victim

The victims have a distinctive colour and are heated allowing a range of detection methods to be used. At least 50% of the victims are also fitted with transponders. These are represented by highly reflective tape on the victim. You have a choice of using reflected light (partial), IR temperature (all) or even a pixycam.

- The best approach under most conditions is to use IR temperature.
  - Works for all victims
  - Reliable and fast
  - Some signal processing is required
- If using light the main problem is that the value for reflected light intensity is also a function of the distance that the sensor is away from the wall and therefore constant wall following distance is critical

This is how you read the temperature sensor that you have named "sensor".
The raw value is stored as a tuple in 100ths of a degree.
You need to extract the number and divide it by 100 to get °C

```
71      temp = sensor.read('TARGET-C') # read temperature
72      Target = temp[0]/100 # convert reading to C
73
74      Colour = light.color() # checks the colour
75
76      if Target < victimT:
```

We are comparing the temperature to a variable that we have defined (around 30 works well)

Finding the Exit tile

- Record the start colour in a variable
- Make sure that you get off the start tile
- Put in another nested if … then to notify your success and turn the robot off.

```
# Write your program here.
ev3.speaker.beep()
wait(500)
start_colour = Color # remember start colour
ev3.speaker.beep()
left_motor.dc(base_speed) # get off start tile
right_motor.dc(base_speed)
wait (2000)


Colour = light.color() # checks the colour

if Color == start_colour: # found the exit tile
    left_motor.dc(0)
    right_motor.dc(0)
    ev3.screen.clear()
    ev3.screen.print("EXIT")
    ev3.speaker.say("Exit found")
    ev3.light.on(Color.ORANGE)
    wait (2000)
    left_motor.dc(50) # move fully onto tile
    right_motor.dc(50)
    wait (500)
    left_motor.dc(0)
    right_motor.dc(0)
    wait (2000)
    ev3.speaker.play_file(SoundFile.CHEERING)
    wait (2000)
    break # stop the program
else:
```

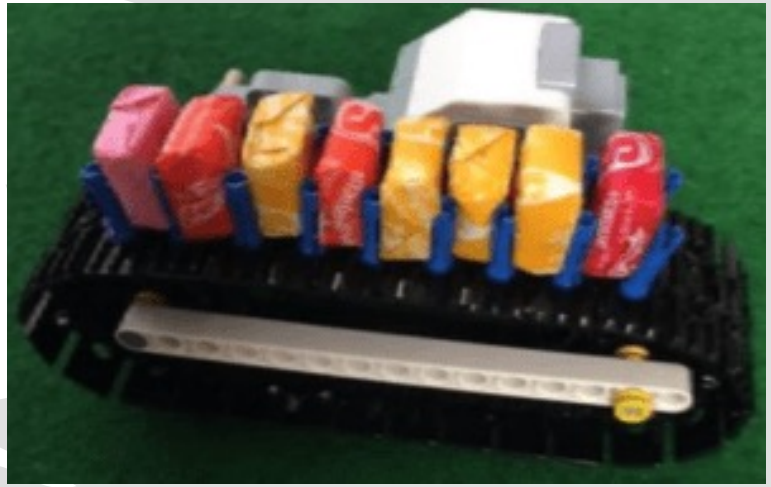
Other cool stuff

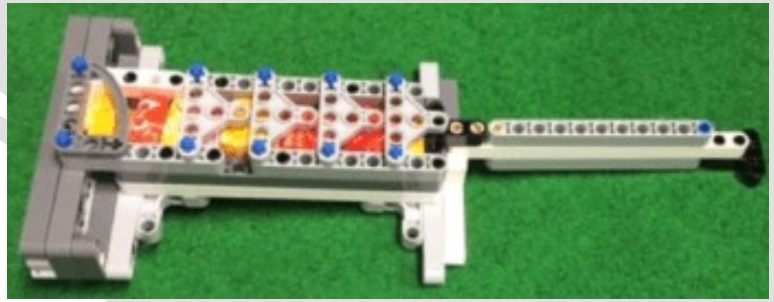
- Displaying info and text on the screen is easy
- Your robot can talk! Try out the different voices and accents
- It is a good idea to have your robot start on a button push. This gives you time to position it exactly as you want.
- While waiting to start display the sensor values to spot any problems early

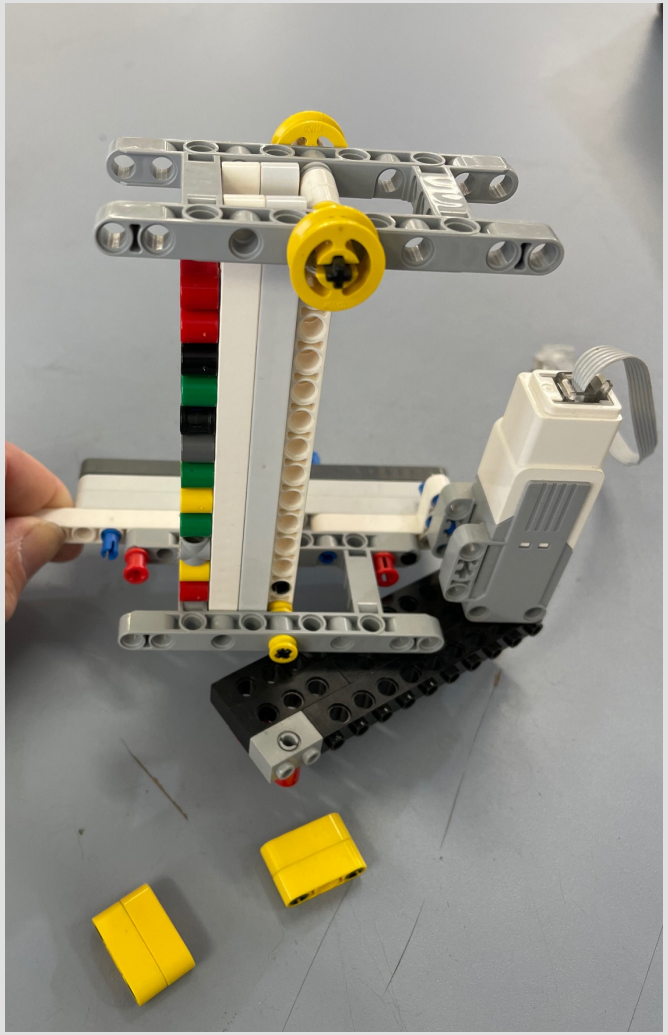
```
ev3.screen.print("HELLO")
ev3.speaker.say("Hello")  # the robot can talk !!!!
wait(1000)
# the following loop displays the sensor values on theh robot.
# allows you to check that everything is working
# will loop until any button is pressed
while not ev3.buttons.pressed(): # looping until button pressed
    ev3.screen.clear()
    ev3.screen.print("WAITING")
    distance = side.distance()
    ev3.screen.print("Side =",distance)
    ev3.screen.print("Front =",front.pressed())
    temp = sensor.read('TARGET-C')
    Target = temp[0]/100
    ev3.screen.print("Target =",Target)
    Colour = light.color()
    ev3.screen.print("Color =",Colour)
    wait(500)
    ev3.screen.clear()
```

Rescue Packages and Dispensers

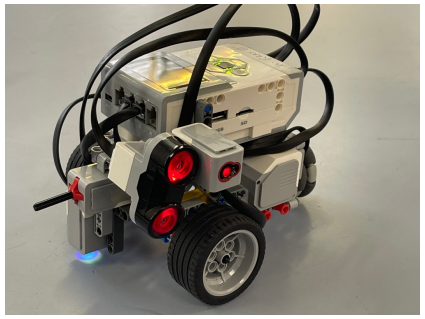
- The design of the packages and dispensers is totally up to you.
- Packages must be at least 0.5 cm^3 with a maximum of 12. You can use M&Ms, gum packages, Lego pieces, laser cut tokens etc.
- Lots of Lego-based solutions including columns, conveyors etc.
- Make sure that your dispenser leaves the package on the same tile as the victim or no points!





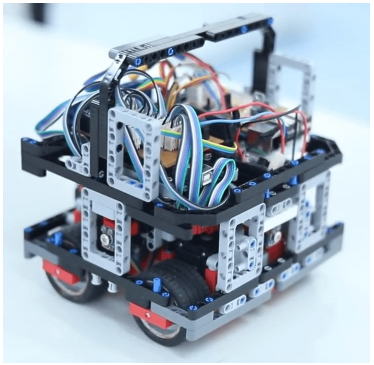


The Next Level



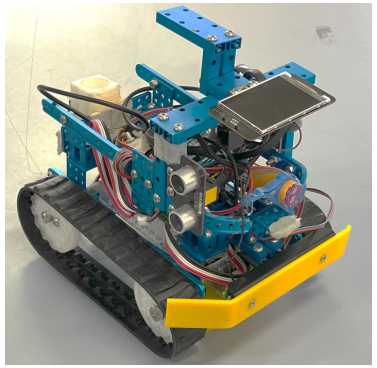
EV3

- Standard Software
- Standard Sensors



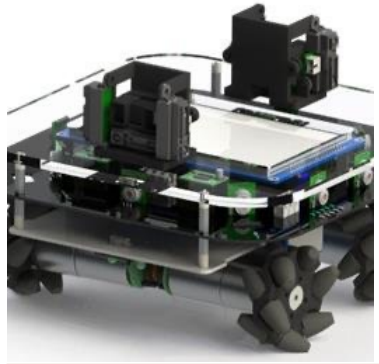
Advanced EV3

- Micropython programming
- Non Lego Sensors (including temperature)
- Servos
- Cameras



Microcontroller

- Arduino or similar
- Makebot robots or similar
- Wide Range of sensors
- Cameras
- Either "real time" or mapping



SLAM Robot

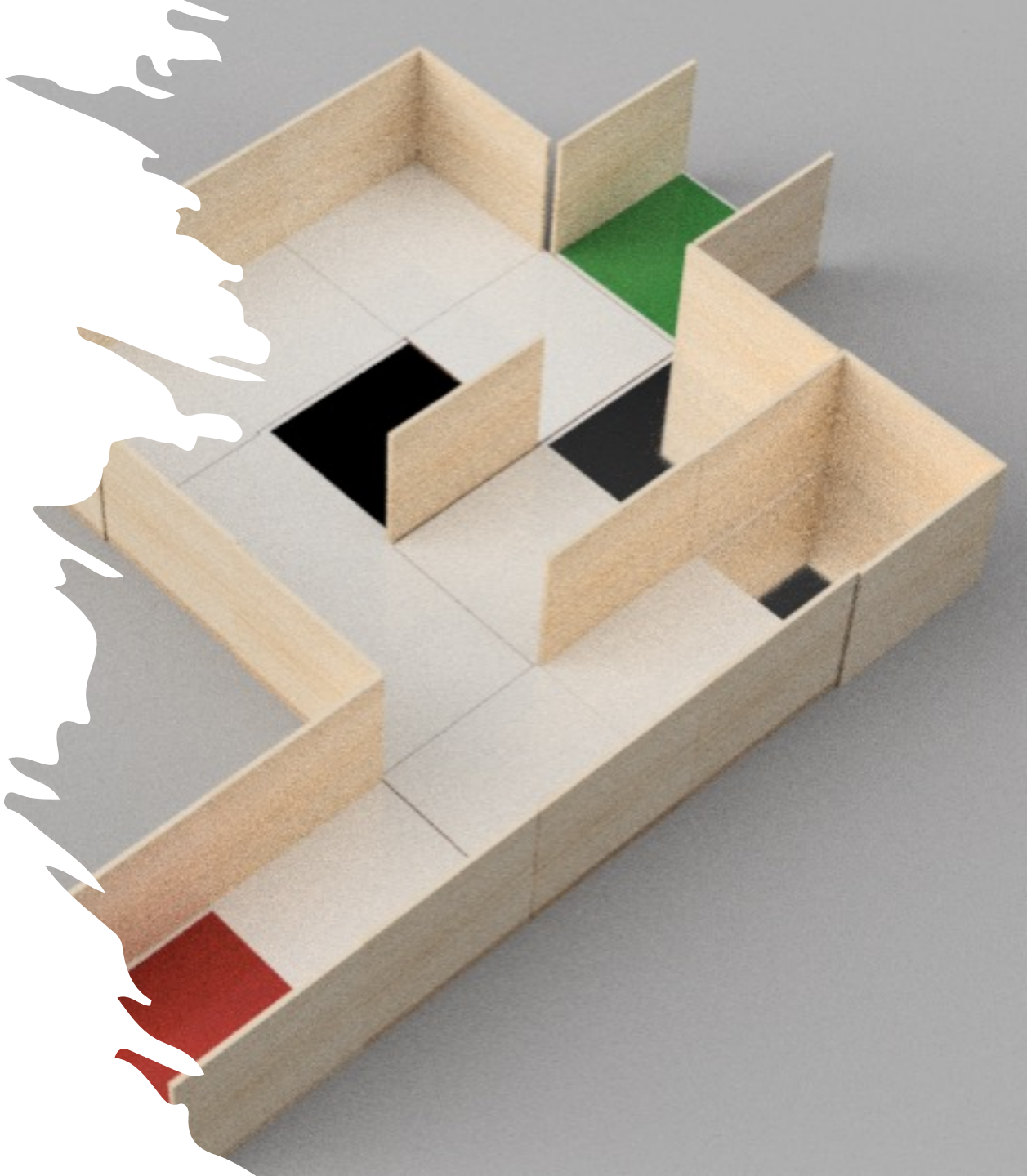
- Raspberry Pi or similar
- Arduino or other microcontrollers
- Visual victims
- Mapping algorithms

RCJA Competitions (<u>all</u> can be competitive)

RCJI Competition (can be evolutionary changes from RCJA)

Building the Maze Elements

- Use the Maze Full Field Cutting Layout (link on website).
- There is a small error in the Ramp.
- 2 Sheets of structural ply 9mm 2400X1200mm
- 2 Sheets White Melamine 16mm 2400X1200mm
- < $200
- Cut to size, glue and nail together.
- Makes the following
- 10 X Blanks,17 X L-shape, 15 X corner, 5 X U-shape, 5 X Dead end, 1 X Ramp
- We are developing a Training Maze Set
- You can always setup a wall and some turning points in the interim.

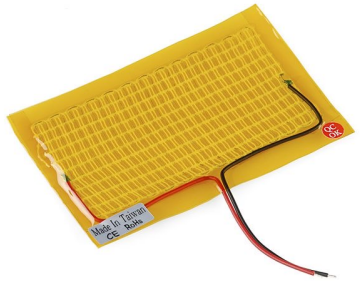


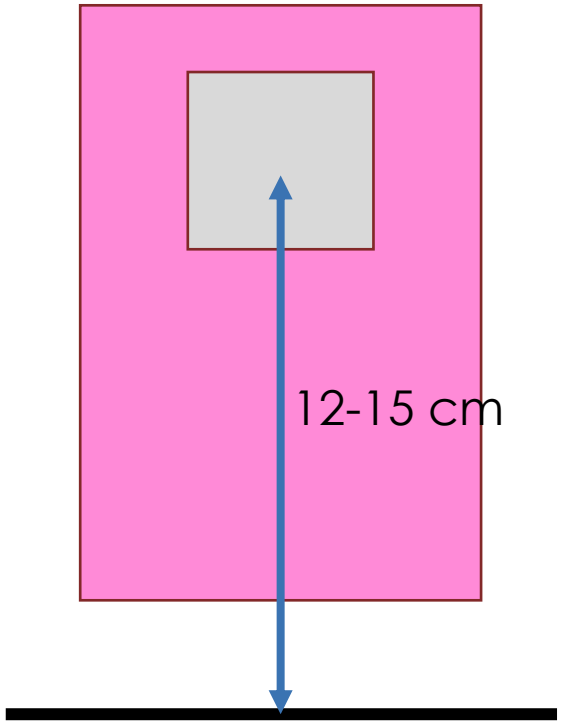
Maze building for noobs

- The timber can be purchased for around $200.
- Allwood Timber Supplies can provide a pre-cut full set of maze elements for $480 (excluding delivery).
  - Contact Matt Albanis
    ALLWOOD TIMBER SUPPLIES
    9350 9611
- Remember that all tiles have a 300mm X 300mm overall size and that the walls are 9mm thick.
- A base for a one-sided tile is then 300mm X 291mm, a U-shaped is 300mm X 282mm etc. Check before you nail and glue.
- The black and coloured squares can be easily made from some black card cut to about 280mm X 280mm
- For the silver checkpoints cover a square with the aluminium tape

The Victims

- Heat not needed for basic EV3 robots but can be included using heater pad powered by 5V supply.
- You will need some
  - Fluro tape
  - Aluminium tape
- Both are available from Jaycar but there are potentially other stockists such as Bunnings and Officeworks
- Tape up a bit of ply, cardboard or corflute with a rectangle of 100mm wide by 130mm high with the fluro tape
- Attach a 50mm square of the aluminium tape as shown as shown
- Position on wall of maze with the reflective square 12-15 cm from the maze floor level
- Solder up heating pad to USB cable and connect to supply.





A Final Word

- Maze Rescue is a fun competition
- Just like the real world it is full of variability and even the fanciest robot will have some bad rounds
- Try to design a reliable and consistent robot
- It is a point scoring competition – not a race.
- Practice, practice, practice
- Don't forget the obstacles and debris – they make a huge difference.
- ***READ THE RULES***.

Important Links

Everything you need will be accessible through the RoboCup junior Australia web site.

I can be contacted at neil.gray@robocupjunior.org.au

- https://www.robocupjunior.org.au/
- https://www.robocupjunior.org.au/rescue-maze/
- https://www.robocupjunior.org.au/wp-content/uploads/2021/02/Official-2021-RCJA-Rescue-Maze-Rules-NGTC.pdf
- https://www.robocupjunior.org.au/wp-content/uploads/2021/02/Maze-Full-Field-Cutting-Layout.pdf