SASTA OLIPHANT SCIENCE AWARDS

SOUTH AUSTRALIAN SCIENCE TEACHERS ASSOCIATION

Prize Winner

Programming, Apps & Robotics

Year 7 - 8

Dhiaan Patel
Ethan Suresh

The Heights School

ROWE SCIENTIFIC PTY. LTD. www.rowe.com.au

Government of South Australia
Department for Education

SASTA South Australian Science Teachers Association

Win the Big Cycling
Getaway valued
at $12,250.


Capacitive Soil
Moisture Sensor V2.0.0

SASTA OLIPHANT SCIENCE AWARDS

SOUTH AUSTRALIAN SCIENCE TEACHERS ASSOCIATION

Computer Programming, Apps & Robotics

Dhiaan Patel & Ethan Suresh

The Heights School

SASTA South Australian Science Teachers Association

Government of South Australia
Department for Education

ROWE SCIENTIFIC PTY. LTD. www.rowe.com.au

Entry Description:

The SoilXplorer is an automated farming robot. The robot moves around on a rail track that needs to be laid out before use but it is more efficient in the long run for the SoilXplorer to move around quickly. The robot relies on the Husky Lens module, April tags(a type of fiducial tags) soil moisture sensor and a water pump to identify the plants and measure the surrounding soil moisture to determine whether to water it and how much it needs to water the plant/crops.

Purpose:

The SoilXplorer main purpose is for farming and plant maintenance on a smaller. The version we have right now is a 40% scale version of the final robot. The robot is placed on rails so that wheels won't damage crops and valuable soil and it saves battery. This helps with battery efficiency, requiring little traction and torque to get going. This saves time for the robot instead of it damaging itself trying to get the robot out of a hole or mud patch that it might have driven into. The robot not only saves time, but it saves energy and manpower on the fields where the farmers work. One challenge for usage could be setting up the rail tracks and setting up the Fiducial tags to make sure the robot identify the plants. The robot saves time, money and energy.

Scientific elements shown:

HUSKYLENS

The SoilXplorer uses a HuskyLens sensor device to detect April tags (A specific version of a fiducial tag for this assembly) using an advanced camera and AI system module called a Huskylens.

The HuskyLens provides a small and compact camera system to strap on the robot without causing much balancing issues and change in initial design features. This means the robot won't have to many issues with the centre of gravity and keeping itself balanced while it operates.

The HuskyLens uses an AI sensor to identify a familiar object with a digital bounding box so that it can identify the objects or fiducial tag's precise location and perform the required action for that object or fiducial tag. A microchip is also installed in the HuskyLens to identify the changing size of the object (if it is an object) depending on if the robot gets closer to it or further from it.

CAPACITIVE SOIL MOISTURE SESNOR

The Capacitive Soil Moisture Sensor is a sensor device used to measure soil moisture to adjust the amount of water it adds into the soil for the plant to take.

The Capacitive Soil Moisture Sensor is a useful sensor because it doesn't corrode over time due to water exposure meaning it has a longer lasting use time and wont have to be replaced regularly. The sensor also takes pride in its high accuracy detections so the amount of water that we give the plant is as precise and compatible as possible so that the plant can live and last as long as possible.

The Capacitive Soil Moisture Sensor uses a rather unique system. Instead of passing a direct current through the soil and water, the Capacitive Soil Moisture Sensor acts as a variable capacitator and lets the surrounding soil and water act as the dielectric. Since water has a higher dielectric constant than dry soil or air. When the soil gets wetter, the dielectric constant increases therefore the capacitance of the sensor increases which means the soil moisture is higher.

THE OVERALL DESIGN:

The overall design is equipped with multiple sensors and motors to move it efficiently along the track while being stable and compact. The robot runs on an Arduino Mega system. It utilizes 2 different sensors. The first being a Moisture sensor to detect moisture. Then next is a HuskyLens that is built to detect fiducial tags (April tags for our design). It also comes with a water pump to pump out water onto the crops.

The robot is optimized to be fast, capable and efficient but simple and easy to use.

THE CODE

The robot is programmed using the Arduino programming language (based on C++). The code is split into clearly defined sections: library imports, pin definitions, settings and tuning values, plant profiles, serial input functions, motor control functions, sensor functions, tag centring, calibration, setup, and the main loop. Each section has a specific job, making the code easier to read, debug and improve.

Libraries & Imports

The first thing the code does is import the HuskyLens library.

```
#include "HUSKYLENS.h"
```

A library is a collection of pre-written code that makes it easier to control hardware. The HUSKYLENS.h library contains functions that allow the Arduino Mega to communicate with the HuskyLens AI camera through Serial1 (UART). The library provides ready-made functions for detecting April Tags, reading their ID numbers and locating their position on the camera screen.

Pin Definitions

These lines assign names to each physical pin connected to the robot's hardware. This makes the code much easier to read and means that if the wiring changes, only one number needs to be updated rather than every line of code.

```
#define DRIVE_ENA   2

#define DRIVE_IN1   22

#define DRIVE_IN2   23

#define ARM_ENB     3
```

```
#define ARM_IN3    24
#define ARM_IN4    25
#define PUMP_PIN   8
#define SOIL_PIN   A0
```

DRIVE_ENA controls the drive motor speed using PWM (Pulse Width Modulation), while DRIVE_IN1 and DRIVE_IN2 control the direction of the drive motors. ARM_ENB, ARM_IN3 and ARM_IN4 perform the same task for the arm motor. PUMP_PIN switches the water pump on or off, and SOIL_PIN reads the capacitive soil moisture sensor connected to the Arduino's analogue input.

Settings & Tuning Values

Several constants are used to control how the robot behaves. These values can be changed without modifying the rest of the program, making it easier to fine-tune the robot.

```
#define DRIVE_SPEED      180
#define CENTRE_SPEED     120
#define ARM_SPEED        200
#define ARM_DOWN_MS      1200
#define ARM_UP_MS        1200
#define SOIL_DRY         700
#define SOIL_WET         300
#define WATER_STEP_MS    500
#define SCREEN_CENTRE_X  160
#define CENTRE_TOLERANCE  20
#define MAX_PLANTS       10
```

These settings control the robot's speed, the amount of time the arm moves, the moisture sensor calibration values, the watering time, and how accurately the robot must align itself with an April Tag before stopping. Storing these values as constants makes the robot much easier to adjust without changing the main program.

Plant Profiles

Rather than storing information for only three fixed plants, the robot creates a profile for each plant during calibration.

```
struct PlantProfile {
  int tagID;
```

```
  char name[32];
  int targetMoisture;
  int freqHours;
  unsigned long lastServicedMs;
};
```

A struct is a custom data type that stores several pieces of related information together. Each plant profile stores the April Tag ID, the plant's name, its ideal soil moisture level, how often it should be checked, and the last time it was serviced. These profiles are stored in an array so the robot can manage up to ten different plants.

Serial Input Functions

Before the robot begins watering plants, it asks the user to enter information through the Serial Monitor.

```
String waitForUserInput();
int waitForUserNumber();
int askForNumberInRange(...);
```

These functions allow the user to enter plant names, moisture levels and watering frequencies. The program also checks that any numbers entered are within a valid range before accepting them. This prevents incorrect settings from being stored and makes the calibration process more reliable.

Motor Control Functions

These functions control the movement of the robot and keep the main program easy to read.

```
driveForward();
driveLeft();
driveRight();
lowerArm();
raiseArm();
startPump();
stopEverything();
```

Each function controls the L298N motor driver by setting the appropriate pins HIGH or LOW while using PWM to control motor speed. Separate functions are also used to raise and lower the sensor arm and turn the water pump on or off. Using dedicated functions reduces repeated code and makes the program easier to understand.

Sensor Functions

The robot uses both a soil moisture sensor and the HuskyLens AI camera to collect information.

int readSoilMoisture();

int readVisibleTag(int &tagXPosition);

The soil moisture function takes five readings and averages them before returning a value. Averaging multiple readings reduces electrical noise and improves accuracy. The HuskyLens function searches for visible April Tags and returns both the tag's ID and its position on the screen, allowing the robot to identify each plant.

Tag Centring Function

Before checking the soil moisture, the robot must accurately position itself in front of the correct plant.

bool centreOnTag(int targetTagID)

This function repeatedly checks the April Tag's position on the HuskyLens screen. If the tag appears too far left or right, the robot slowly turns until the tag is within a set tolerance of the screen centre. This feedback system allows the robot to position itself accurately before lowering the moisture sensor. If the robot cannot centre on the tag within 15 seconds, the function stops and reports that it was unsuccessful.

Calibration Function

The robot includes an interactive calibration system that runs when it first starts.

runCalibration();

During calibration, the user enters the number of plants and registers each one individually. The robot detects the plant's April Tag, centres itself, asks the user for the plant's name, target soil moisture and watering frequency, then stores this information in the plant profile array. This allows the same robot to work with many different gardens without changing the code.

Setup Function

The setup() function runs once when the Arduino is powered on.

void setup()

The function starts serial communication, configures all output pins, stops every motor for safety, connects to the HuskyLens camera using Serial1, selects the April Tag recognition algorithm, and then runs the calibration process. If the camera cannot be detected, the program stops immediately to prevent the robot operating without its vision system.

Main Loop

The loop() function contains the main operation of the robot.

```
for (int i = 0; i < numPlants; i++) {
  servicePlant(plants[i]);
}
```

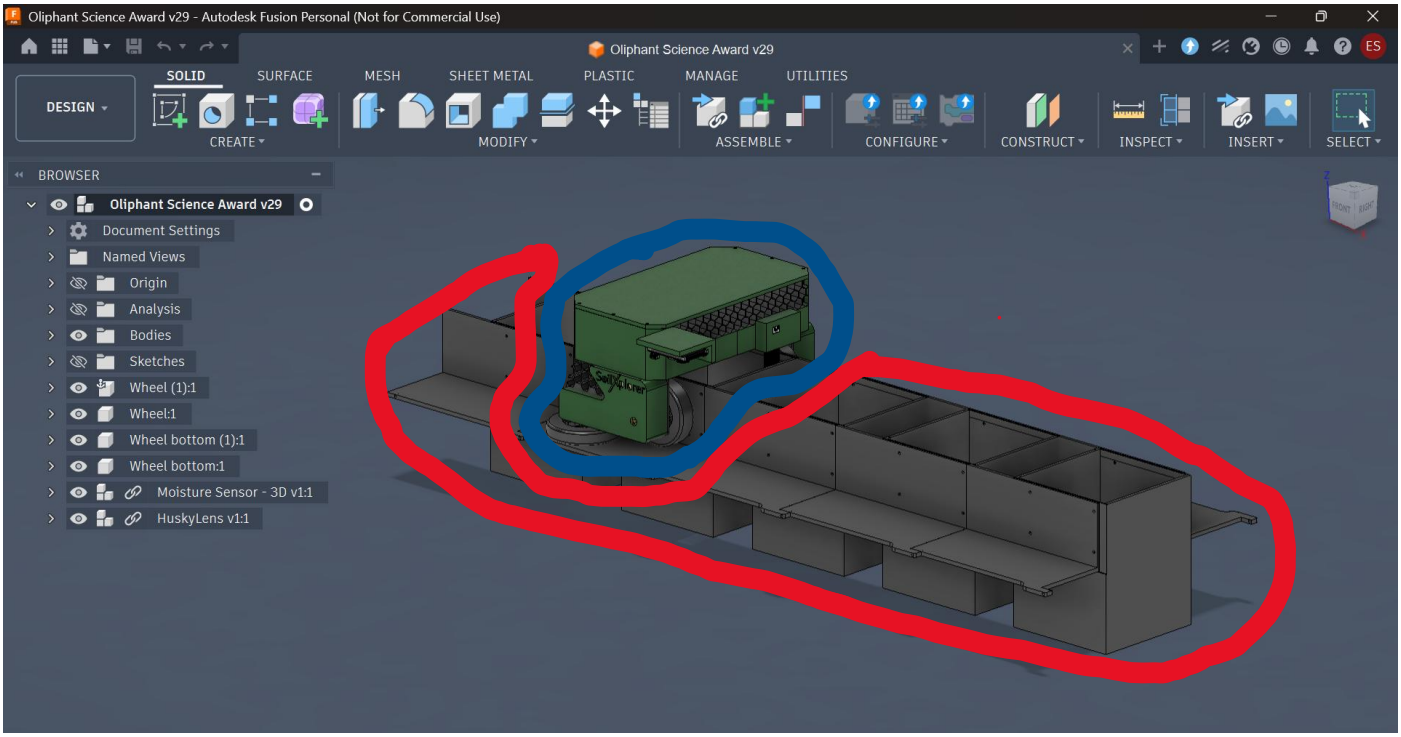
The robot visits every registered plant one at a time. For each plant, it checks whether enough time has passed since the previous inspection. If the plant is due, the robot finds the correct April Tag, centres itself, lowers the moisture sensor into the soil, measures the moisture level, raises the sensor, and decides whether watering is required. The watering time is automatically calculated based on how dry the soil is. After all plants have been checked, the robot waits 30 minutes before starting another inspection cycle.

Hardware

The Model Consists of Two Main Parts: The Plant Pot and The Moving Assembly

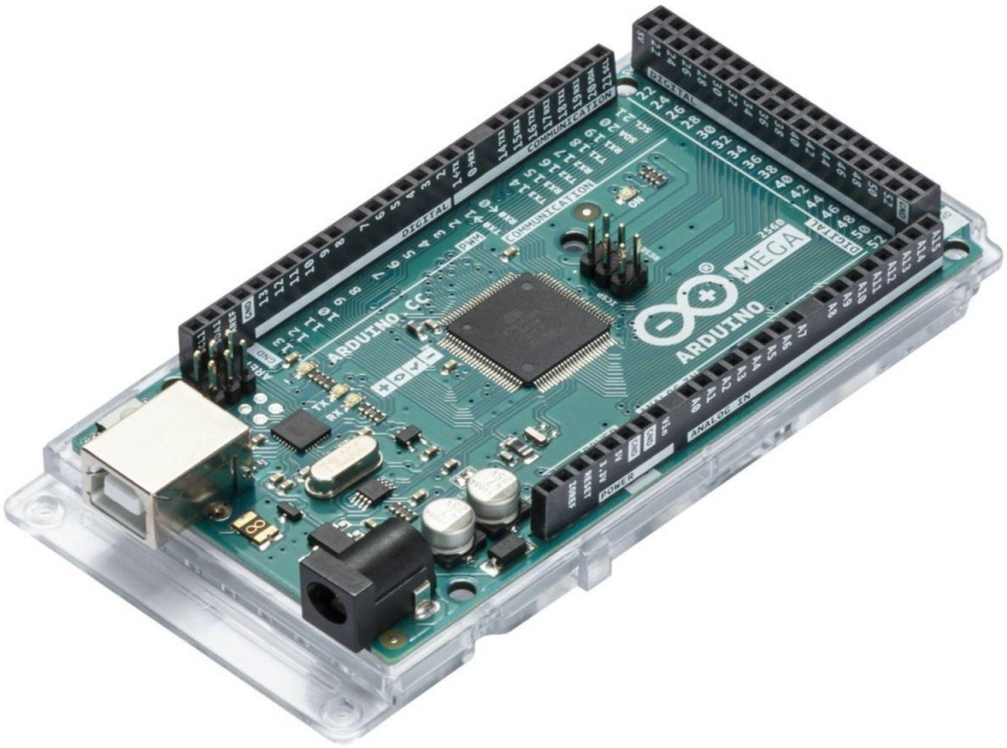
(The photos of the robot are the early stages of the robot and isn't wired, assembled or soldered)

The renders of the physical robot in Fusion 360:



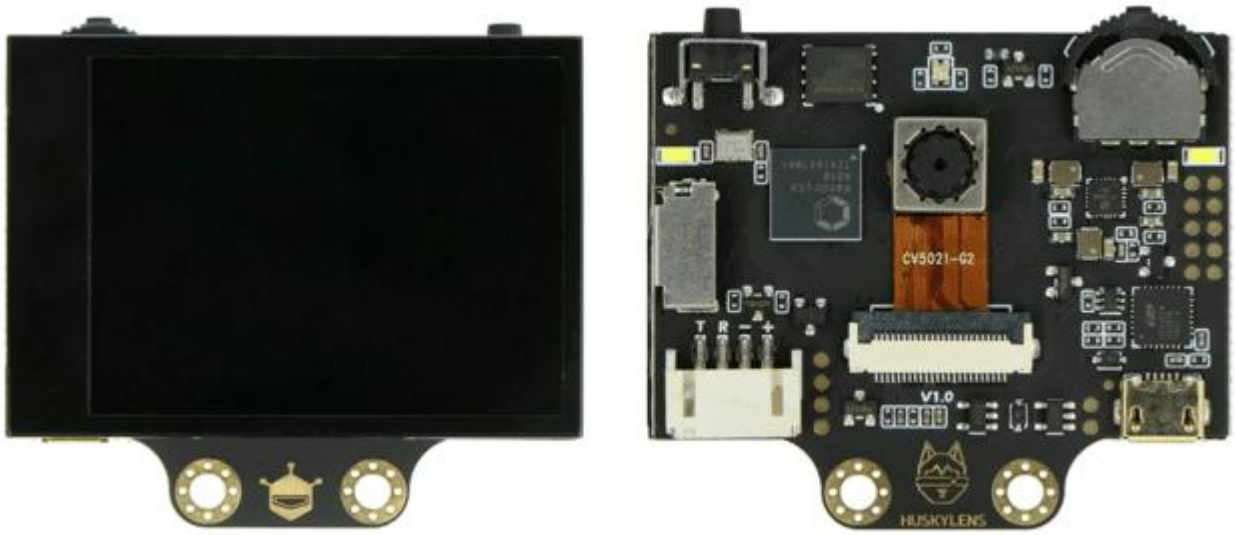
Red = Plant Pots

Blue = Moving assembly that waters, measures soil moisture and reads the April tags



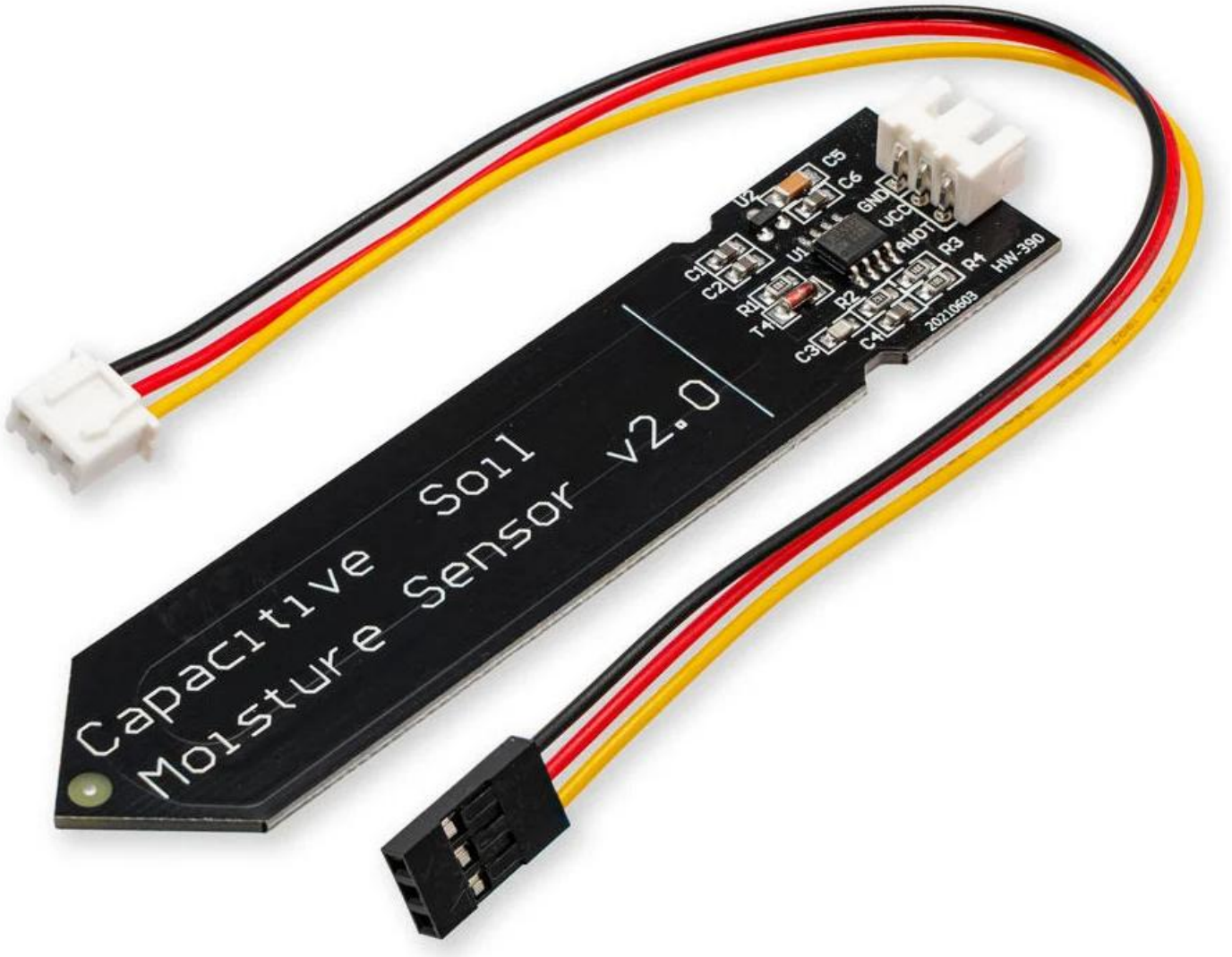
Arduino Mega 2560 r3

The Arduino Mega is the brain of our project and controls everything. It is a programable module that controls the entire robot and makes sure it functions correctly.



Huskylens

The Huskylens is the eyes for the robot and detects different tags for different types of plants. It detects the tags then tells the Arduino and then the Arduino reacts according to that tag. ‘React’ meaning watering the plant.



Capacitive Soil Moisture Sensor

The Capacitive Soil Moisture Sensor reads the value of the soil moisture and tells the Arduino. Then the Arduino waters based of that and what the soil moisture is supposed to be from the information given through the tags.

Motors

The motors turn a certain distance that we have tuned because they are DC meaning they are either on or off. We had to tune to a number of seconds the motors were turned on for to get the exact distance of each plant. The motors turn and then the program tells what the Arduino must do next according to the list of steps in the program.

Acknowledgement of any support or help:

All external help that we received during our time in this competition was received from our school (The Heights School) who kindly supplied us with the required parts needed to construct the robot (Arduino Mega, HuskyLens, motors, Water pump, etc) and some of the tools needed to assemble the robot. Our school also helped us 3d print certain parts of the robot.

References:

During the process of creating our robot we unfortunately did not keep track of many of the sites because we finished and created this report only a few days before the deadline. We are still able to list some of the main contributing websites or apps that helped us along the way.

For building, assembly and design of the robot we used:

Autodesk Fusion360

We used Fusion360 to create the model and design of our robot. It is a creation platform where we could create the parts that we needed to complete our robot. (I couldn't provide a link to the app so here is the link to the website to download the app) https://www.autodesk.com/products/fusion-360/personal#faq

Oliphant Science Awards Webpage

We used the Oliphant Science Awards webpage for the information for this tournament. https://www.oliphantscienceawards.com.au

Tinkercad

We used Tinkercad to mostly program our robot through using block coding and 'translating' it to c++ using the in-built text and blocks feature. This would reduce the amount for c++ we needed to know helping us make our code. https://www.tinkercad.com

For ordering the parts we used

Amazon https://www.amazon.com.au

Core Electronics https://core-electronics.com.au/?gad_source=1&gad_campaignid=625924959&gclid=CjwKCAjw6f3RBhApEiwAMaCqWZRMFMxjzH8EYPwLJU7y06I380hA71PSmtK3o0DlOIUnuiPf2myABhoCLMAQAvD_BwE

Jaycar
https://www.jaycar.com.au/?srsltid=AfmBOopZCrKxMqjEyX2vmCLPHFcrJVRbO4QagoKTzsU0lunMzjyAwXjt