

**Internal Assessment Resource**

**Achievement Standard Digital Technologies 91375:** Implement advanced interfacing procedures in a specified electronic environment

**Resource reference:** Digital Technologies 2.48 v2

**Resource title:** Recycling Robot

**Credits:** 3

Achievement	Achievement with Merit	Achievement with Excellence
Implement advanced interfacing procedures in a specified electronic environment.	Skilfully implement advanced interfacing procedures in a specified electronic environment.	Efficiently implement advanced interfacing procedures in a specified electronic environment.

**Student instructions****Introduction**

This assessment activity requires you to construct an autonomous mobile robot that is capable of following a course marked out by a black line on a white background. The robot has to be capable of finding a soft drink can in a designated area 400x400mm, picking the can up and depositing it on a platform 40mm high and 100mm long located at one edge of the designated area.

Due Date: 23 March 2018

The specifications for the autonomous mobile robot are:

- the robot must be capable of following the course specified.
- the chassis is to be constructed from 4mm pvc or acrylic sheet
- the wheels, motors and gears may be LEGO components (but could be other proprietary products)
- the line sensors are to be constructed from the LDRs and LEDs
- the robot must be equipped with a gripper capable of lifting up a soft drink can containing 40g of ballast. This gripper can be powered by either a d.c. motor or a servo motor
- the robot must be able to detect the platform on which the can is to be placed, approach the platform and place the can on the platform so that it does not fall over when released.

The specifications for the 3 interfaces are:

- an interface that allows a microcontroller to manage independently and reliably the speed and direction of a pair of small 9V d.c. electric motors. Any EMI generated by the motors should not be allowed to interfere with the reliable functioning of the robot.

- an interface that allows a microcontroller to reliably distinguish between green black and white markings on a vinyl sheet over which the robot is driving. The main sensor is to consist of a small light dependent resistor wired up with a resistor to form a voltage divider. The sensor does not need to make more than 500 measurements per second.
- an interface, which allows a microprocessor to make a sound when it has completed each task (e.g. when picks up and deposits can).

Relevant health and safety practices must be followed in the making of the product.

## Prerequisite tasks

Read and absorb the specifications above.

If you have had no previous experience in using some of the actuators, sensors, or with the programming environment in which you are working then, practise extensively using these resources until you are confident you know exactly how they work and how to get the best performance out of them.

Check that you are aware of relevant Health and Safety practises and know how to follow these when making your product.

## Task

In constructing your robot there are three main interfaces for which you must develop functional models. They are:

- one to manage the motors
- one to manage the sensors that detect the line that the robot will be following
- one to manage the output of sound at appropriate times.

Each of these interfaces will require development work in both hardware and software.

As you develop these interfaces you must keep records - photos and notes - of the work you are doing. These records will be used to write a report, which will be submitted with the robot for the assessment of this standard.

Your report must show evidence (see the appendices for example) that you have:

- used the electronic components provided to produce a sensor that can interact with the environment. This means the sensor can accurately detect the colour of the surface it is looking at and produce a voltage that the software can use to determine that colour
- used the electric motors provided so that the software on the robot can reliably control the speed and position of the robot
- used the microcontroller to enable the robot to provide useful information to the programmer thereby allowing the performance of the robot to be improved
- written, tested and debugged well-structured, clearly annotated, readily understandable software to manage the interface between the robot's processor and the sensors and actuators it controls
- You should provide evidence of modification of the sensor subsystems and actuator subsystems, which substantially improves the quality and the way the robot works/gathers data.

Typically your evidence should show that you have been doing a selection of the following things:

- selecting the best type and value of component
- selecting the best arrangement of components
- modifying hardware input and or output parameters
- modifying software parameters.

You can meet the above requirements by including in your written evidence:

- schematic circuit diagrams for any electronic circuits you develop as part of an interface
- photos of any interfaces that you construct
- a brief description explaining the role of each interface
- brief descriptions of any testing and debugging procedures you undertook to substantially improve the operations of the interfaces
- photos and explanations of any modifications you make to any of the interfaces to substantially improve their performance
- screenshots of any software that interfaces with either the actuators or the sensors on your robot. These screenshots should show that your software is well structured, clearly annotated and readily understandable.

Hand in your robot and your report.

## Resources

### ***Resource 1: Sets of techniques that are likely to be required in producing evidence for this standard include:***

Design of circuit schematics.

Selection and assembly of electrical connectors.

Techniques to reduce unwanted effects of electromagnetic interference.

Data logging techniques - maybe including averaging or other methods to improve reliability.

Construction techniques to fabricated mechanical components of interfaces.

### ***Resource 2: Health and safety***

Students must be able to use all machines and processes safely and in accordance with the Technology Departmental policies on Health and Safety.

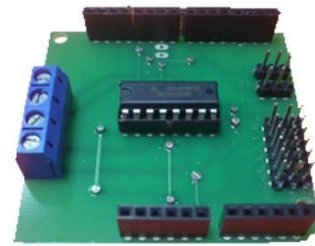
Examples of these are:

- ensure sleeves rolled up when using any machines
- wear safety glasses when machining
- ensure all machine guards are fitted and working before starting machines
- ensure machines are turned off before using any measuring instruments
- ensure anti-static mats are used when working with circuits
- ensure equipment is safely stored

## Appendix A: Student Evidence for Achieved

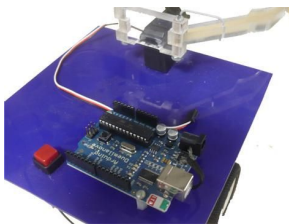


LCD interface



actuator interface

The student used the actuator interface to control the two motors that drove his mobile robot, and also to control the servo motor that opened and closed the grippers. The lcd interface was used to obtain readings from the light sensors that enabled the robot to follow the line marked on the course course.



This photo shows the microprocessor mounted on the robot and the grippers mounted on the servo motor that operated them.



This photo shows the LED and the LDR mounted on a printed circuit board. This is the main part of the light sensor interface. The LED shines light on to the terrain under the robot and the LDR receives the reflected light. One of the resistors forms a voltage divider in partnership with the LDR thus generating a signal, which is dependent on the amount of light reflected, and is also able to be read by the robot's microprocessor.

The C-language code below shows that the student has written software that interfaces with the robots sensors and actuators. However, although the code is well formatted, the absence of any annotation makes it difficult to determine what the code is doing, and even harder to troubleshoot any errors that may be present.

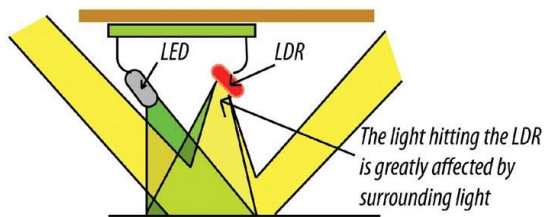
```
void setup()
{
}
```

```
void loop()
{
```

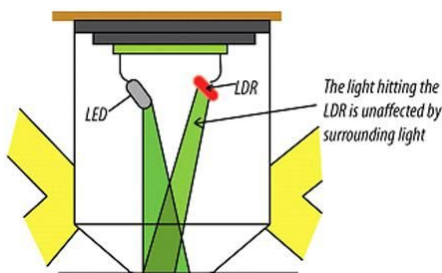
```
if (analogRead(3) > 350)
{
    digitalWrite(6,HIGH);
    digitalWrite(7,LOW);
}
else
{
    digitalWrite(6,LOW);
    digitalWrite(7,LOW);
}
if(analogRead(right_sensor) >375)
{
    digitalWrite(4,HIGH);
    digitalWrite(5,LOW);
}
else
{
    digitalWrite(4,LOW);
    digitalWrite(5,LOW);
}
}
```

## Appendix B: Student Evidence for Merit

Sensor:



The initial sensor design was a simple circular circuit board that contained a LED shining light on to the ground, which reflected off the different coloured surfaces at different intensities, which was picked up by an LDR. The robot could then do line following based on the digital reading that the LDR produced. However, this design was greatly affected by the surrounding light conditions, so much that in a darker room, the programmed reading for a white area could be read as a black line in a well-lit room. No matter how precise I made the programming, this variation could not be overcome.



I changed the design of the sensor so that the circuit board was attached to a plastic piece, which became one lid of an open-ended plastic cylinder. The other end was fitted with another lid, which had only a small hole in it. The effect of this design was that the light from the LED shone down, but only the small area of light that hit the hole reflected back to the LDR to increase the sensitivity to the black line. The cylinder blocked almost all surrounding light out which made the readings far more stable and the calibration programme could easily account for this.

### Assessor Comment:

The above sample shows evidence of a student modifying an interface to improve the way it works. In this case the nature of the modifications was geometrical and mechanical, but nevertheless they achieved the purpose of improving the quality of the data that was being gathered by the robot's microprocessor.

The C-language code below shows how the student used software that interfaced with the data provided by the sensors and with the actuators. In this case the data coming in from the sensors was used to control the motors and keep the robot following the black line. The code is well formatted and adequately annotated. However, as the Excellence example shows, it could be made significantly more straightforward to read than is the case here.

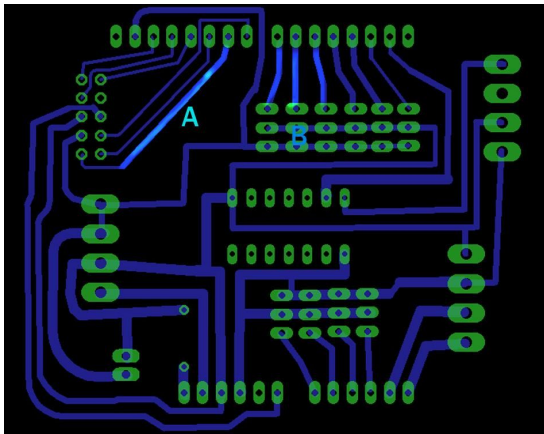
```

void setup()
{
}

void loop()
{
    if(analogRead(3) > 350)           //left sensor is detecting white background
    {
        digitalWrite(6,HIGH);        //left motor is attached to pins
        digitalWrite(7,LOW);         // 6 and 7 on microprocessor
    }
    else
    {
        digitalWrite(6,LOW);         //both pins low turns motor off.
        digitalWrite(7,LOW);
    }
    if(analogRead(right_sensor) >375) //if right sensor is detecting white background
    {
        digitalWrite(4,HIGH);        //right motor is attached to
        digitalWrite(5,LOW);         // pins 4 and 5 of microprocessor
    }
    else
    {
        digitalWrite(4,LOW);         //both pins low turns motor off.
        digitalWrite(5,LOW);
    }
}

```

### Appendix C: Student Evidence for Excellence



The image shows the tracks for an interface board, which connected both a pair of motors and also a group of sensors to the microcontroller. During testing it was found that either the motors or the sensors would work perfectly, but as soon as the motors and sensors were run together, the data entering the sensors was corrupted. Eventually it was realised that track A was transmitting high frequency pulses to control the speed of one of the motors, and the three tracks highlighted at B were transmitting data from the sensors into

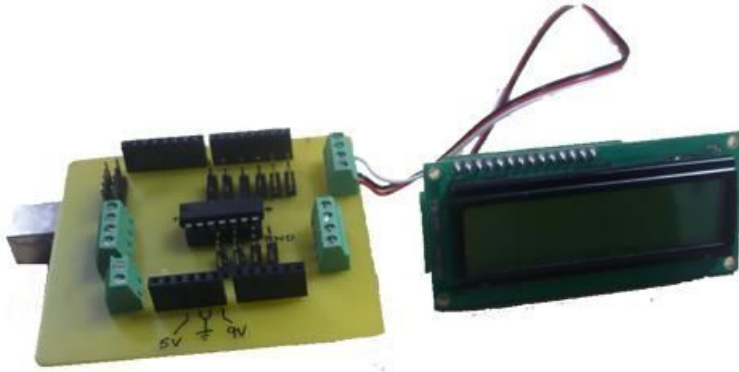
the microcontroller. The rapidly changing pulses on track A were inducing random voltage in the tracks at B and contaminating the data coming in from the light sensors. The solution was to shift the light sensor inputs to the opposite side of the board from the motor tracks. Magnetic fields attenuate rapidly with distance and the problem was solved. Further testing showed that the data stream coming in from the sensors was now completely unaffected by what was happening with the motors.

**Assessor Comment:** This example provides evidence of debugging an interface and also of making significant modifications, which substantially improve the performance of the system.



This image shows a modified d.c. motor interface that is a significant improvement on the one pictured in the Achieved notes. The integrated circuit, which manages the motor, is capable of delivering twice as much current as the original. Also the speed control features of the integrated circuit have been implemented in this circuit so the robot can now control the speed of its motors. This interface has also been fitted with suppression diodes to reduce the possibility of electromagnetic noise generated by the motors interfering with the operation of the robot's computer. In addition the motor interface has now been mounted well away from the main computer with a ribbon cable used to connect the two boards. This again reduces the chance of electromagnetic interference reaching the robot's microprocessor. And finally, a heat sink has been added to the integrated circuit to dissipate the additional heat that is generated by the larger currents that are now flowing.

As described above this interface was tested in conjunction with the robots sensors, and initially produced cross-talk with the sensor inputs. However this was rectified by shifting the sensor inputs to another location, well away from the motor outputs.



This image shows another modification to the interfacing system. In this case the original parallel lcd screen has been replaced by a serial one. The significance of that is that the parallel version used 6 of the microprocessors 10 input pins, which was preventing a number of important sensors from being attached. The serial lcd panel uses only two pins, thus making 4 more available to use for other sensors. The change to a serial lcd panel was also required additional software to be written to manage communication between the lcd panel and the microcontroller.

### ***Example of Well-Structured, Clearly Annotated and Readily Understandable Program Code***

The example below shows some code, which is used to interface the robot's microprocessor to a pair of light sensors and also to a pair of d.c. motors.

Note the use of the labels *left\_sensor* and *right\_sensor* which make the code easier to understand.

Also note the naming of the functions that control the motor. These make the main (LOOP) section of the code extremely straightforward to read and understand.

Note also the use of comments within the code that explain what various lines of code are doing.

Also the indentation of the code each time a new set of brackets open makes it easier to understand the logical structure of the program.

Finally, note the clear separation of the different sections of the code that, again, helps to make it readily understandable.

```

/*****
****

```

In this code fragment, the robot's sensors look at a white background, on either side of a black line.

If the left sensor strays on to the black line, the left motor stops until the sensor returns to the white

zone. Same for the right sensor. If both sensors are on the white zone, the robot moves forwards.

```

*****
*****/

```

```

int left_sensor = 3;    //left sensor is attached to pin 3 on microprocessor
int right_sensor = 4 ;  //right sensor is attached to pin 4 on microprocessor

```

```

void setup()
{
}

```

```

void loop()
{
    if(analogRead(left_sensor) > 350)  //left sensor is detecting white background
    {
        left_motor_forwards();        //so left motor drives forwards
    }
    else

```

```

{
    left_motor_stop();          //otherwise left motor stops
}

if(analogRead(right_sensor) >375)  //if right sensor is detecting white background
{
    right_motor_forwards();      //right motor drives forwards.
}
else
{
    right_motor_stop();          //otherwise right motor stops
}

}

/*****

function definitions
*****/

{
    digitalWrite(4,HIGH);        //right motor is attached to
    digitalWrite(5,LOW);         // pins 4 and 5 of microprocessor
}
/*****/

void right_motor_stop()
{
    digitalWrite(4,LOW);         //both pins low turns motor off.
    digitalWrite(5,LOW);
}
/*****/

void left_motor_forwards()
{
    digitalWrite(6,HIGH); //left motor is attached to pins
    digitalWrite(7,LOW); // 6 and 7 on microprocessor
}

```

```
/******  
void left_motor_stop()  
{  
    digitalWrite(6,LOW); //both pins low turns motor off.  
    digitalWrite(7,LOW);  
}
```