

# **Robokid 2**

## Technical Notes

Jim Herd  
Andy Haston  
June 2022

## Contents

Robokid 2 .....	1
Introduction .....	3
2 System.....	3
2.1 Functional subsystems.....	3
2.2 Technical subsystems .....	4
3 Functional subsystems.....	6
3.1 System management .....	6
3.2 User interaction .....	8
3.2.1 Joystick input.....	8
3.2.2 Display .....	9
3.2.3 Switches/LEDs .....	9
3.2.4 Potentiometers .....	10
3.2.5 Sound output.....	10
3.3 Actuation.....	10
3.3.1 Motors.....	10
3.3.2 Wheels .....	11
3.4 Communication interfaces.....	11
3.4.1 USB interface.....	11
3.4.2 IR link .....	11
3.5 Sensors.....	12
3.5.1 Line sensors .....	12
3.5.2 Bump sensors.....	12
3.5.3 Light sensors .....	14
3.6 System power .....	14
3.7 Miscellaneous .....	15
4.0 Proposals.....	16
Appendices .....	17
Appendix A : Software process structure .....	17

## Figures

Figure 1 General system structure .....	5
-----------------------------------------	---

## Tables

Table 1 Functional subsystems .....	3
Table 2 Technical subsystems .....	4
Table 3 Some Processor options.....	7

## Introduction

This document brings together technical issues in the design of Robokid 2. The success of Robokid 1 means that many of the principles used in its design will be carried through to the new design. The fundamental change will be the use of more modern technology. A second document will look at the wider resource environment relevant to the running of a Robokid 2 project in schools

It should be noted that Robokid 1, and by extension Robokid 2, are aimed at P5/P6/P6 level of primary schools. However, if developed with a programming capability, it could have application in secondary schools.

## 2 System

### 2.1 Functional subsystems

The system is split into a number of functional subsystems

Name	Responsibilities
System management	<ul style="list-style-type: none"><li>• Processor</li><li>• Memory</li></ul>
User interaction	<ul style="list-style-type: none"><li>• USB SNES Gamepad</li><li>• Push buttons</li><li>• Potentiometers<ul style="list-style-type: none"><li>◦ Allows user to input variable parameters<ul style="list-style-type: none"><li>▪ e.g. speed</li></ul></li></ul></li><li>• LED displays</li><li>• LCD display</li><li>• Tone speaker</li></ul>
Actuation	<ul style="list-style-type: none"><li>• DC motors</li><li>• Wheel type</li><li>• Motor sensing</li></ul>
Comms Interfaces	<ul style="list-style-type: none"><li>• USB/FTDI Virtual com port</li><li>• IR</li></ul>
Sensors	<ul style="list-style-type: none"><li>• Line sensors</li><li>• Light sensors</li><li>• Touch sensors</li></ul>
Power	<ul style="list-style-type: none"><li>• Battery</li><li>• Charging</li><li>• Condition sensing</li></ul>
Miscellaneous	<ul style="list-style-type: none"><li>• ??????</li><li>• RGB lighting</li></ul>

Table 1 Functional subsystems

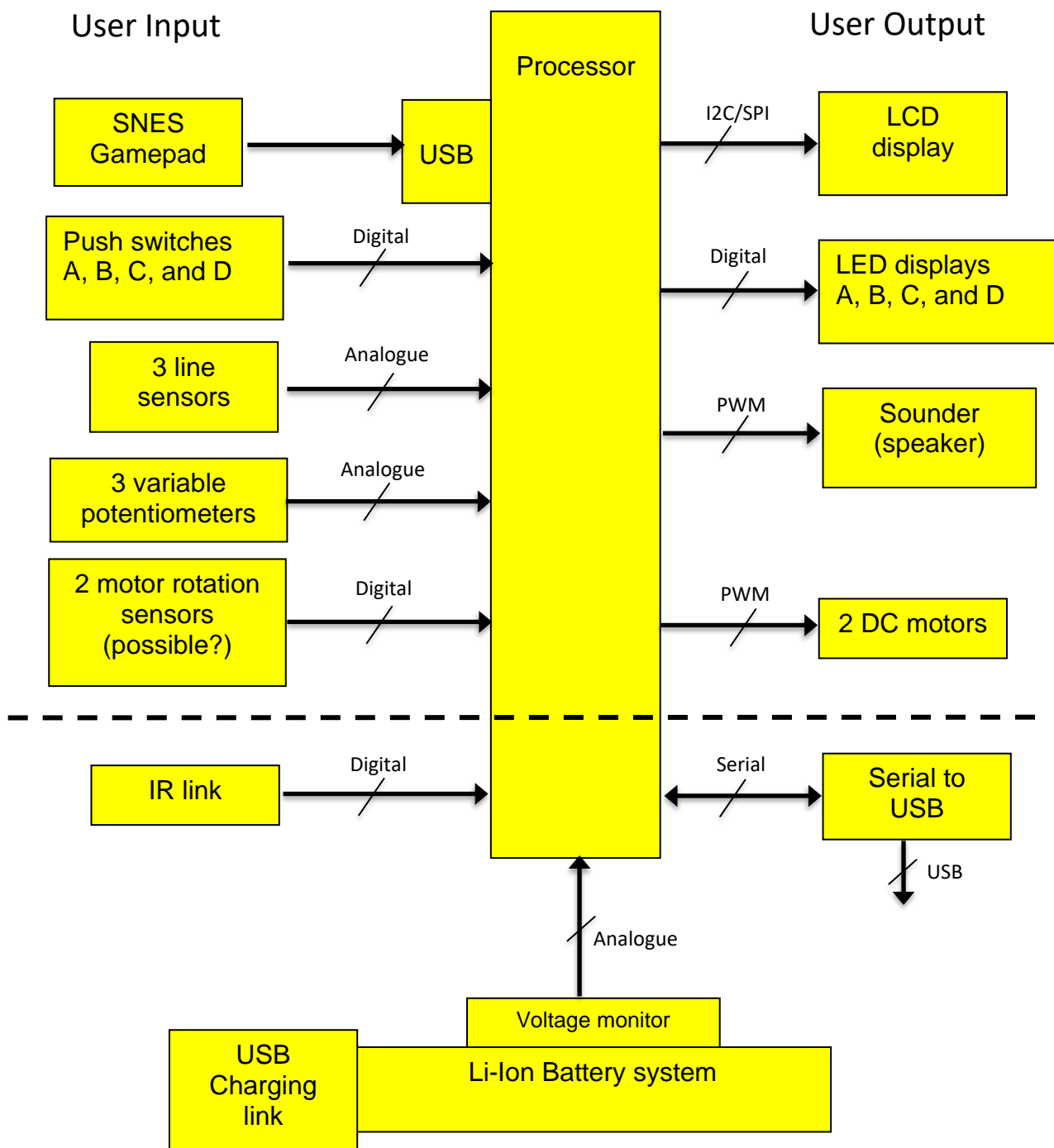
## 2.2 Technical subsystems

The above list of subsystems implies the need for the following technical components.

Name	Responsibilities
Processor	<ul style="list-style-type: none"><li>• Microcontroller</li></ul>
USB	<ul style="list-style-type: none"><li>• SNES Gamepad (USB HID class)</li><li>• Com port (USB CDC class)</li></ul>
Digital I/O	<ul style="list-style-type: none"><li>• Push buttons</li><li>• LED displays</li><li>• IR receiver</li></ul>
Analogue I/O	<ul style="list-style-type: none"><li>• IR line sensors</li><li>• Potentiometers</li></ul>
PWM Output	<ul style="list-style-type: none"><li>• Motor control<ul style="list-style-type: none"><li>◦ H-bridge control</li></ul></li><li>• Sound output<ul style="list-style-type: none"><li>◦ Tone Speaker</li></ul></li></ul>
Interfaces	<ul style="list-style-type: none"><li>• I2C/SPI<ul style="list-style-type: none"><li>◦ LCD display</li></ul></li><li>• Serial<ul style="list-style-type: none"><li>◦ FTDI Virtual Com Port</li></ul></li></ul>
Power system	<ul style="list-style-type: none"><li>• Li-Ion battery<ul style="list-style-type: none"><li>◦ Regulation</li><li>◦ Charging</li><li>◦ Monitoring</li></ul></li></ul>

**Table 2 Technical subsystems**

Figure 1 shows the connections between the subsystems and the central microcomputer.



**Figure 1 General system structure**

## 3 Functional subsystems

### 3.1 System management

The processor provides the “smarts” of the system, but does not need to be a top-range device as all likely activities are not intrinsically complex. Clearly, cost will be an issue and we will look to use a cost-effective device.

The following table gives some options and their pros and cons

Device	Notes
Raspberry Pi zero	<p>Bottom of the range of Raspberry Pi range of computers.</p> <p>Pros</p> <ul style="list-style-type: none"><li>• Powerful for its size.</li><li>• cost range : mid to high (relative to other options)</li><li>• Wireless capability (see later discussion)</li><li>• Commitment to longevity of production</li></ul> <p>Cons</p> <ul style="list-style-type: none"><li>• Volume supply issues (not related to semiconductor supply issues)</li><li>• Needs SD card (could be lost)</li><li>• Needs to be shutdown correctly to prevent SD card corruption</li><li>• No analogue input</li><li>• Not available as a chip, therefore would have to use the board.</li></ul>
Raspberry RP2040	<p>Recent Arm based microcontroller chip from the makers of the Raspberry Pi range of computers.</p> <p>Pros</p> <ul style="list-style-type: none"><li>• Modern chip with full range of required interfaces</li><li>• Available as a chip.</li><li>• High clock rate with two cores</li><li>• British designed chip</li><li>• Cost range : low</li><li>• Good documentation and software tools</li><li>• Commitment to longevity of production</li><li>• Family only has one device – easy decision.</li></ul> <p>Cons</p> <ul style="list-style-type: none"><li>• No wireless (see later)</li><li>• Restricted number of analogue input channels</li><li>• Relatively new device</li><li>• Family only has one device – no choice.</li></ul>

ST STM32	32-bit Arm based microcontrollers from ST Microelectronics  Pros <ul style="list-style-type: none"> <li>• Good range of devices. Plenty of choice.</li> <li>• Cost range : low to mid</li> <li>• Commitment to longevity of production</li> <li>• I have used one of these in another project.</li> </ul> Cons <ul style="list-style-type: none"> <li>• Unsure about documentation and support</li> <li>• Extensive range of devices</li> <li>• Unsure about supply situation</li> </ul>
ESP32	Espressif developed microcontroller with embedded wireless and Bluetooth.  Pros <ul style="list-style-type: none"> <li>• Very capable device</li> <li>• Embedded wireless TCP/IP stack (see later)</li> </ul> Cons <ul style="list-style-type: none"> <li>• Complex to use as a chip (need to use module)</li> <li>• Cost range : mid</li> <li>• Unsure about documentation and support</li> <li>• Unsure about supply situation</li> </ul>
Arduino	Pros <ul style="list-style-type: none"> <li>• Heavily used system in one-off applications.</li> <li>• Good support</li> </ul> Cons <ul style="list-style-type: none"> <li>• Little used in volume applications</li> <li>• Cost range : mid (probably have to use a module rather than a chip)</li> </ul>
Others	????????????????????

**Table 3 Some Processor options**

In a low performance, low quantity application like Robokid 2, there is unlikely to be a killer reason for choosing one device over another. The easiest option to discount is the Raspberry Pi Zero. The power on/off and SD card issues would not work well in a school classroom environment.

The list makes a number of references to wireless capability. The questions to be addressed : is it worthwhile paying the extra premium to have this capability? On a cursory consideration it would seem justifiable for the expenditure of and extra £2/£3 pounds. The ESP32 would seem an ideal solution. However, on consideration of the environment in which Robokid 2 will operate – the primary school classroom, there are a number of factors to consider

1. The majority of primary school teachers do not have a background in STEM/IT. In general they are very interested in the area, but have a limited knowledge.
2. The majority of primary schools do not have an in-house IT department to fix/solve any issues that may arise with either Wi-Fi or Bluetooth networking.

3. Schools are unlikely to have open Wi-Fi networks and education authority IT departments are unlikely to grant access to tens of Wi-Fi devices without significant assessment on their part. In fact, I would think that access would be denied. I'm not sure Heriot-Watt IT would allow this!

For these reasons, I would suggest that Wi-Fi/Bluetooth should not be part of a schools robot which would remove the ESP32 device from consideration.

That leaves RP2040, STM32 and Arduino. I have used an Arduino in a small one-off project and it worked well, was quick to code and had lots of libraries. Beyond this, I have no experience of higher volume applications. My impression is that it is not heavily used for such applications. Most of the details about Arduino focus on boards rather than chips. For this reason I will remove the Arduino from my list.

Just RP2040 and STM32 left. Probably little to choose between these two. I am using an STM32 chip for another robot project which needs a CAN interface. However, I decided to use the RP2040 for my Robokid 2 test system. It is fast, 133MHz, dual core, plenty of memory, and cheap. As a UK designed chip, there may be some publicity traction in its use. From a technical perspective, my initial impressions are good.

<b>Proposal 1</b>	Build test system with the RP2040 chip.
-------------------	-----------------------------------------

## 3.2 User interaction

### 3.2.1 Joystick input

Robokid 1 used a crude in-house designed “joystick” with two 3-position switches. This worked and had the benefit of simplicity and cheapness, but at the expense of flexibility.

More recent processors have USB interfaces that give the opportunity to use the processors as a host to a USB joystick or gamepad. The RP2040 has such an interface.

Initial experiments have been conducted with a SNES gamepad, shown here

This is a relatively cheap option with a total of 12 digital buttons which will give more flexibility in the set of “joystick” activities.

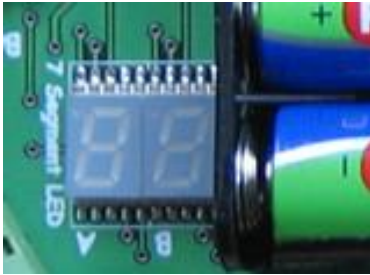


<b>Proposal 2</b>	Build test system to allow the use of a USB joystick/gamepad
-------------------	--------------------------------------------------------------



### 3.2.2 Display

The robot system needs some way of displaying information to the user. Robokid 1 used a simple two character 7-segment display which can show a passable representation of most the alphabet. The system incorporates a scrolling system to display simple text.



We need a better way to communicate with the user but within the constraints of cost and size. A good compromise option is the SSD1306 OLED display

Although quite small (0.96 inch in the horizontal direction) the OLED technology gives a bright and clear result. They have a display area of 128 by 64 pixels and access is at the pixel level. Thus, all user messages need to be built as a map of pixels before transfer to the display. These are very common devices and are supported by a number of open-source C libraries. The interface can be either I2C or SPI.



For the demo system I have modified a library to maximise the speed of update which provides a display area of 4 rows of 15 characters with vertical scrolling capability to show bigger messages. Both I2C and SPI interfaces have been tested.

<b>Proposal 3</b>	Use SSD1306 to provide the display
-------------------	------------------------------------

### 3.2.3 Switches/LEDs

In Robokid 1, input from the user to set operating modes was with a set of four push button switches. This proved ideal, with each switch accompanied by a LED, which would be lit to highlight an active switch. In the videos and literature, it was useful to specify Robokid operating modes as a string of ABCD button press values. For example, sequence ADDDAA (shortened to A3D2A in the documentation) set Robokid into a line follow mode.

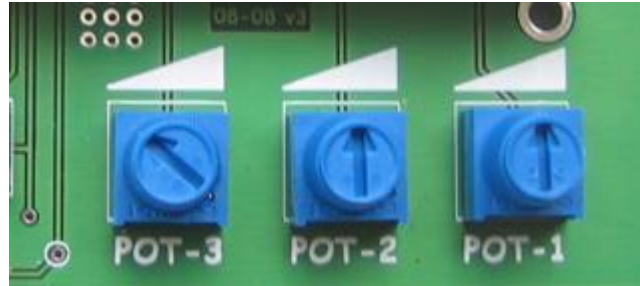


I would suggest that the same would be used on Robokid 2.

<b>Proposal 4</b>	Include 4 push switches and associated LEDs.
-------------------	----------------------------------------------

### 3.2.4 Potentiometers

Switches can only give a Yes/No inputs. Sometimes we want the user to input a value, e.g. required speed. Without attachment to a computer, we need a different approach. In Robokid 1 we used three variable resistors, called potentiometers (POTs). The processor can read the potentiometer value and relate it to the quantity being specified. These potentiometers are linear therefore the user is presented with a well-behaved input.



Perhaps, Robokid 2 could use LEDs to indicate active POTs.

<b>Proposal 5</b>	Include 3 potentiometers for parameter input and consider inclusion of LEDs to indicate active POTs
-------------------	-----------------------------------------------------------------------------------------------------

### 3.2.5 Sound output

Some form of sound output provides good feedback to inputs and events. For example, acceptance of a button press, or detection of a black line when in bump mode. Given that there may be as many as 13 robots in a classroom, speech output could be easily drowned out by the general class noise. Loud buzzes would seem a better option.

There was a piezo buzzer on Robokid 1, but because it was not particularly loud and was placed on the underneath of the circuit board, it was not very effective. Better placement and the use of an amplifier feeding a PCB speaker should give a more effective system.

<b>Proposal 6</b>	Include amplified PCB speaker giving tones and buzzes.
-------------------	--------------------------------------------------------

## 3.3 Actuation

### 3.3.1 Motors

The primary purpose of the robot is to allow interesting movements under manual or automatic control. Two DC motors will provide this capability. These are the main mechanical components of the system and, as such, need to be reasonably reliable. In the design of Robokid 1 I took the decision, based on some preliminary testing of cheap motors with some pupils, to buy a good quality DC motors with metal gears and a bearing on the output shaft. Given that we recycled robots term by term, and year by year for five years, I do not remember there being any failures. I paid about £4 (incl VAT) per motor.

<b>Proposal 7</b>	Use good quality DC motors
-------------------	----------------------------

### 3.3.2 Wheels

Robokid 1 used a cheap friction fit plastic wheel which worked surprisingly well. Robokid 2 could use a similar technology.

However, it maybe worthwhile considering an alternative. One area where the wheel was deficient was the measurement of wheel position and speed. If volume allowed, it would be possible to design and manufacture a custom injection molded wheel with

- Wheel shaft hole with keyed design to fit keyed shaft of DC motor
- Molded structures to aid IR wheel rotation sensing
- Design to allow use of O-ring to enhance wheel grip
- Robotarium branding on the wheel rim

Clearly, such an approach is based on volume.

<b>Proposal 8</b>	Review wheel design/purchase
-------------------	------------------------------

## 3.4 Communication interfaces

Although I indicated earlier that I would not recommend the use of Wi-Fi/Bluetooth communication, I included two communication method in Robokid 1.

### 3.4.1 USB interface.

It is possible to get inexpensive chips to allow a USB connection to a UART serial port of a microcontroller. These devices allow a computer to open a bidirectional serial communications port to the microcontroller. This will allow us to investigate the use of Robot programming systems (e.g., Scratch, Blockly, etc.) to produce sequences that can be downloaded and executed on the Robot.

<b>Proposal 9</b>	Include USB-to-Serial capability
-------------------	----------------------------------

### 3.4.2 IR link

Communication via an infra-red beam is the dominant technology for TV controllers. The technology has the advantage of providing a simple on-way wire-free directed communications link. An IR link test system was built a couple of years ago to connect a PC to Robokid 1. This was connected to the PC audio out and was very successful. The approach would not need the installation of any drivers as the entire software can be run from a web page on the PC. With a bit of care on the users part interference between different groups can be minimized. The demo system used a simple coding scheme based on a simple modulation scheme. Of course, the downside is that the communications in one way only. However, the download link is sufficient to transfer a pupil generated program to the robot. In quantity the cost per robot would be about £3 – 0.75p for the robot sensor and £2/£3 for the dongle to connect to the audio port and a USB port (for power only).

<b>Proposal 10</b>	Investigate use of an IR link
--------------------	-------------------------------

## 3.5 Sensors

Ways in which a robot can sense its environment allows a greater number of interesting activities that more clearly relate to real-world applications.

### 3.5.1 Line sensors

Downward facing IR sensors can be used to detect black tape. An array of 2 or 3 sensors can be used to detect the tape and its edges - forms the basis of line following and bump sensing activities. However, the output of these sensors are analogue values that need to be read by an A-to-D converter. The RP2040 only has 4 analogue input channels therefore there will be a need to extend the number of channels. As we are dealing with low grade analogue values, we can use a cheap CD4051 CMOS analogue multiplexer to feed a single channel on the RP2040.

Proposal 11	Incorporate set of IR line sensors and extend the number of analogue channels
-------------	-------------------------------------------------------------------------------

### 3.5.2 Bump sensors

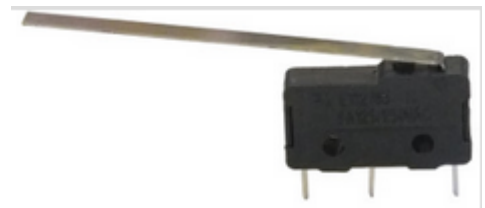
Detecting when the robot hits an object can give some interesting activities. Options include contact and non-contact detection. Options include

Type	Notes
Contact	Microswitches
Non-contact	IR, laser, ultrasonics

#### 3.5.2.1 Contact sensing

Microswitches provide the majority of contact sensors. The level type shown in the picture would provide




Building these onto the Robokid 2 circuit board could present reliability issues. They could easily be broken in normal system activities that did not involve the bump facility.



Proposal 12	Provide a facility to allow microswitches to be attached to the robot for particular activities.
-------------	--------------------------------------------------------------------------------------------------

### 3.5.2.2 Non-contact sensing

More sophisticated, therefore more expensive, sensors are available to detect an object before the robot makes contact.

Technology	Notes	
Ultrasonic	<p>Quite an old technology. Bulky and gives poor results when used close to the ground (problem with reflections). Also rather slow.</p> <p>As Robokid is a low vehicle, ultrasonic ranging would not be viable.</p>	
IR beam	<p>IR emitter and receiver are packaged together and based on the geometry can calculate the distance to an object. There are a range of devices for different ranges.</p> <p>Quite expensive and bulky.</p>	
Laser	<p>A more recent ranging technology is based on laser time-of-flight measurement. These are very compact, being designed for the mobile phone market.</p> <p>A primary supplier is ST Microelectronics based in Edinburgh.</p> <p>Might be interesting to gauge their interest in being a partner to the project.</p> <p>For information, they provided, for free, 1600 3.3v regulators for the Robokid 1 project.</p>	 <p><b>VL53L5CX</b> 4th generation FlightSense™</p>

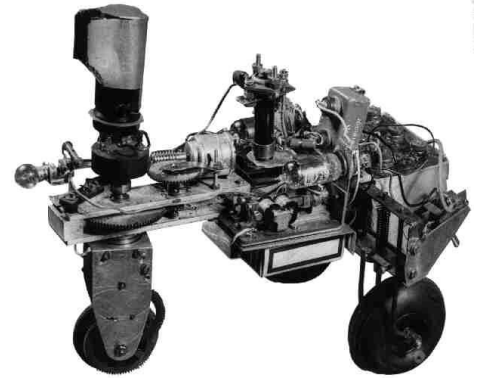
<b>Proposal 13</b>	Investigate the potential use of time-of-flight sensors for object detection.
--------------------	-------------------------------------------------------------------------------

### 3.5.3 Light sensors

Having an ability to sense light and dark could lead to some interesting behavior activities. Relates back to the early days of robot vehicles, when in 1948 William Grey Walter developed the first light following autonomous robot vehicle.

Would be good to have pupils repeat this experiment.

Activities could use cheap LDR (Light Dependent Resistor) technology which would generate analogue input values.



Proposal 14	Investigate use of LDR input sensors
-------------	--------------------------------------

### 3.6 System power

Robokid 2, like Robokid 1, must be powered by batteries. The main issues to be considered are

- Battery chemistry
- Battery protection
- Battery charging

Robokid 1 used FOUR Nickel-metal Hydride (NiMH) AA batteries. When the decision was made it was relatively easy. NiCad batteries with its “memory” problem was being replaced by the much better NiMH technology that was available at reasonable cost.

However, the decision for Robokid 2 is a little more complex. It would seem that the obvious decision would be to move to a Lithium based technology. The rapid development of mobile phones has driven this technology. The downsides of Lithium based technologies are

1. Battery short can lead to high internal temperatures and could lead to bursting into flame. Therefore, additional battery protection circuitry is essential.
2. Care needs to be taken when charging. Again, specialized circuitry is required.

These issues imply the following

- Battery should be fixed into the robot and pupils should not have access to the unprotected terminals or to be able to remove the unit. This is quite different from Robokid 1 where removing and charging the AA NiMH batteries was part of the groups activity.
- Short circuit protection MUST be built in.
- Charging circuitry MUST be built in.
- As with mobile phones, the voltage source for charging should be a USB connection. However, we may need to consider the provision of an external USB system with multiple outputs to allow up to 11 robots to be charged in a reasonable time.

Proposal 15	Review battery options
-------------	------------------------



### **3.7 *Miscellaneous***

Everything I have discussed so far has had a particular technical reason to be included. However, there may be some items that can be included for fun. I am open to suggestions.

Here's one

- Include some serial addressable RGB lights (WS2812). Could be used to give the robot some nice colour effects.

## 4.0 Proposals

The following is a list of the proposals discussed in the report.

Proposal	Description	Decision	Completion
1	Build test system with the RP2040 chip.		50%
2	Build test system to allow the use of a USB joystick/gamepad		75%
3	Use SSD1306 to provide the display		75%
4	Include 4 push switches and associated LEDs		90%
5	Include 3 potentiometers for parameter input and consider inclusion of LEDs to indicate active POTs		90%
6	Include amplified PCB speaker giving tones and buzzes.		75%
7	Use good quality DC motors		
8	Review wheel design/purchase		
9	Include USB-to-Serial capability		
10	Investigate use of an IR link		
11	Incorporate set of IR line sensors and extend the number of analogue channels		
12	Provide a facility to allow microswitches to be attached to the robot for particular activities.		
13	Investigate the potential use of time-of-flight sensors for object detection.		
14	Investigate use of LDR input sensors		
15	Review battery options *** Important ***		



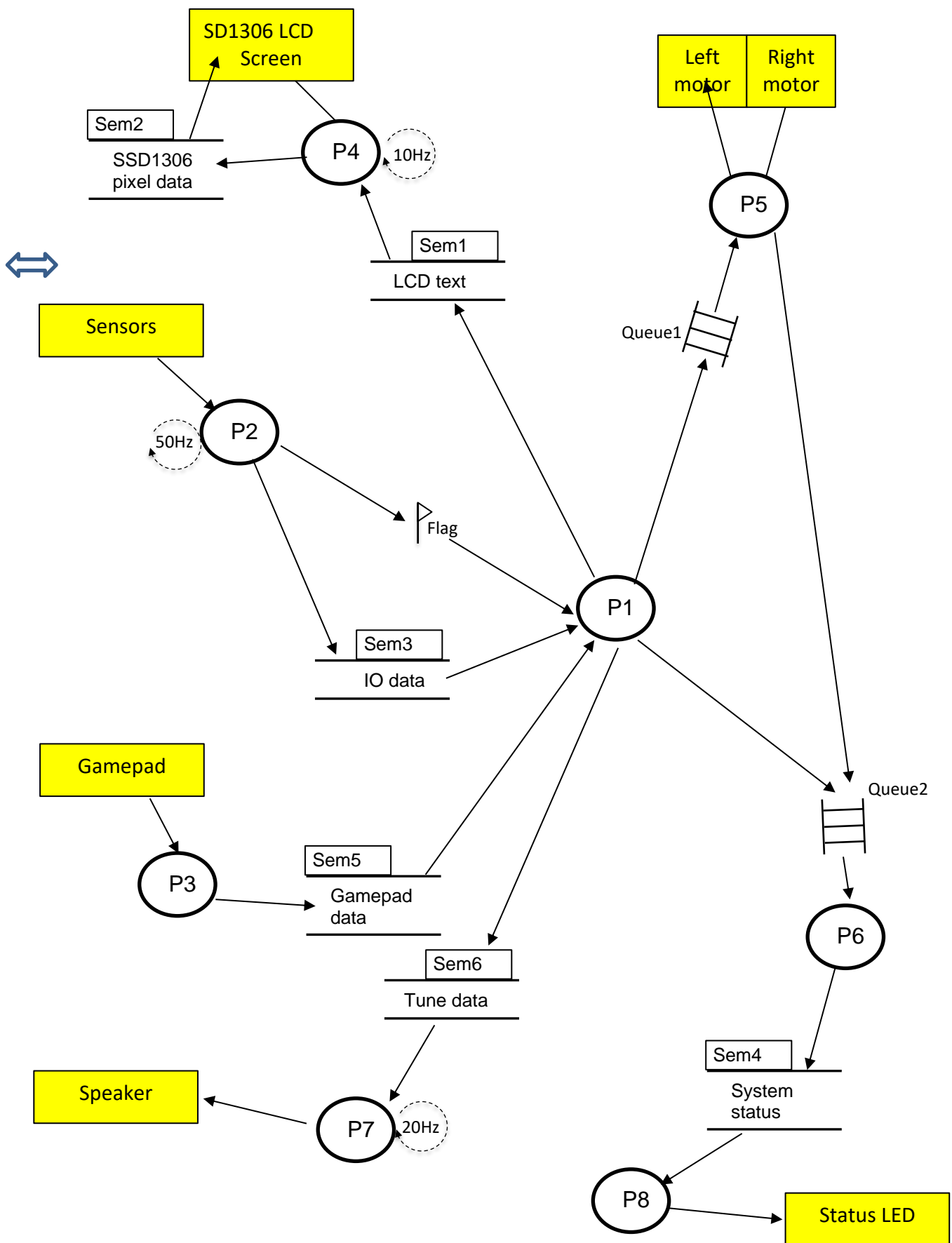
# Appendices

The appendices document more specialized technical aspects that are not relevant to the discussions in the main text.

## ***Appendix A : Software process structure***

The software in Robokid 1 was based on a background/foreground structure with the foreground consisting of an interrupt routine that executed a range of timed activities. Although adequate for the application, it lacked flexibility. With the enhanced compute power of the RP2040 chip (133MHz clock versus 48MHz clock, 32-bit versus 8-bit) I was able to implement the current test software with a real-time operating system – FreeRTOS, which allows the software to be built as a set of cooperating tasks. FreeRTOS is a mature, open source real-time operating system used in many applications.

Structure as of June 2022



Name	Type	FreeRTOS entity names	Notes
P1	Process	Task_Robokid	
P2	Process	Task_read_sensors	
P3	Process	Task_read_gamepad	
P4	Process	Task_display_LCD	
P5	Process	Task_drive_motors	
P6	Process	Task_error	
P7	Process	Task_sounder	
P8	Process	Task_blink_LED	
Queue 1	Queue	queue_motor_cmds	
Queue 2	Queue	queue_error_messages	
Sem1	Semaphore	semaphore_LCD_data	Tasks : P1, P4
Sem2	Semaphore	semaphore_SSD1306_display	Tasks : P4
Sem3	Semaphore	semaphore_system_IO_data	Tasks : P2, P5
Sem4	Semaphore	semaphore_system_status	Tasks : P4, P6, P8
Sem5	Semaphore	semaphore_gamepad_data	Tasks : P1, P3, P4
Sem6	Semaphore	semaphore_tune_data	Tasks : P1, P7
Flag1	Flag	eventgroup_push_buttons	