|  |  |
| --- | --- |
|  | |
| Department: | Computer Science |
| Project: | Wellingborough School Smart Watch |
| Code: | CS-007 |
| Academic Year: | Y12 |

**Overview**

In this project, you will write the code for a Smart Watch. You will use an LCD Display Module, connected to a Raspberry Pi; the Raspberry Pi will provide all of the ‘smart’ capabilities of the Smart Watch; the LCD Display Module is simply a display with an embedded driver. We will use the Serial Peripheral Interface (SPI) standard to send information from the Raspberry Pi to the display controller.

Our Smart Watch will have the following functions:

* Time (12/24 hour switchable)
* Date (UK and US formats)
* Multiple watch faces
* Message alerts
* Location

In addition to running the Smart Watch code on the Raspberry Pi, you can also write a ‘Smart Phone app’ to interact with the Smart Watch, and add some physical buttons to control the Smart Watch’s functions.

You can also come up with an extended set of functions for your Smart Watch, for example, gathering stock prices or flight/train/traffic information from the Internet to display on the watch.

The extended project enables us to add new sensors to the Smart Watch (temperature, humidity) and to display this information.

**Parts**

Raspberry Pi

Waveshare 1.28” LCD Module (<https://www.waveshare.com/wiki/1.28inch_LCD_Module>)

Colour-coded wires x 8

Optional: Breadboard

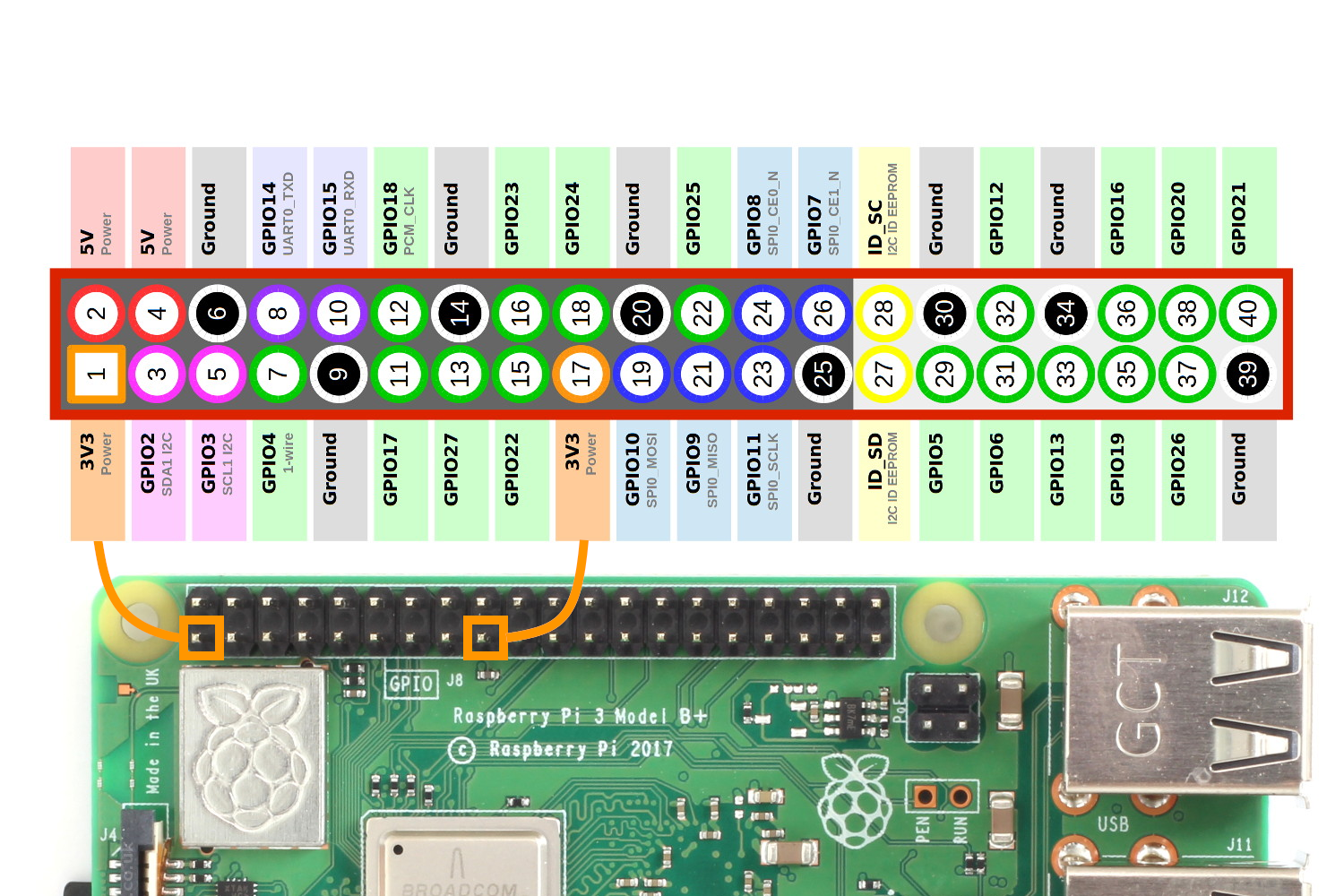
Optional: Push buttons

**Instructions - Hardware**

Connections

|  |  |  |  |
| --- | --- | --- | --- |
| Display label | Meaning | GPIO Header Pin | GPIO Name |
| VCC | Supply (+3.3v) | 1 | 3V3 |
| GND | Ground (0v) | 6 | GND |
| DIN | SPI Data Input | 19 | MOSI |
| CLK | SPI Clock | 23 | SCLK |
| CS | Chip Select | 24 | CE0 |
| DC | Data/Command Select | 22 | P25 |
| RST | Reset | 13 | P27 |
| BL | Backlight | 12 | P18 |

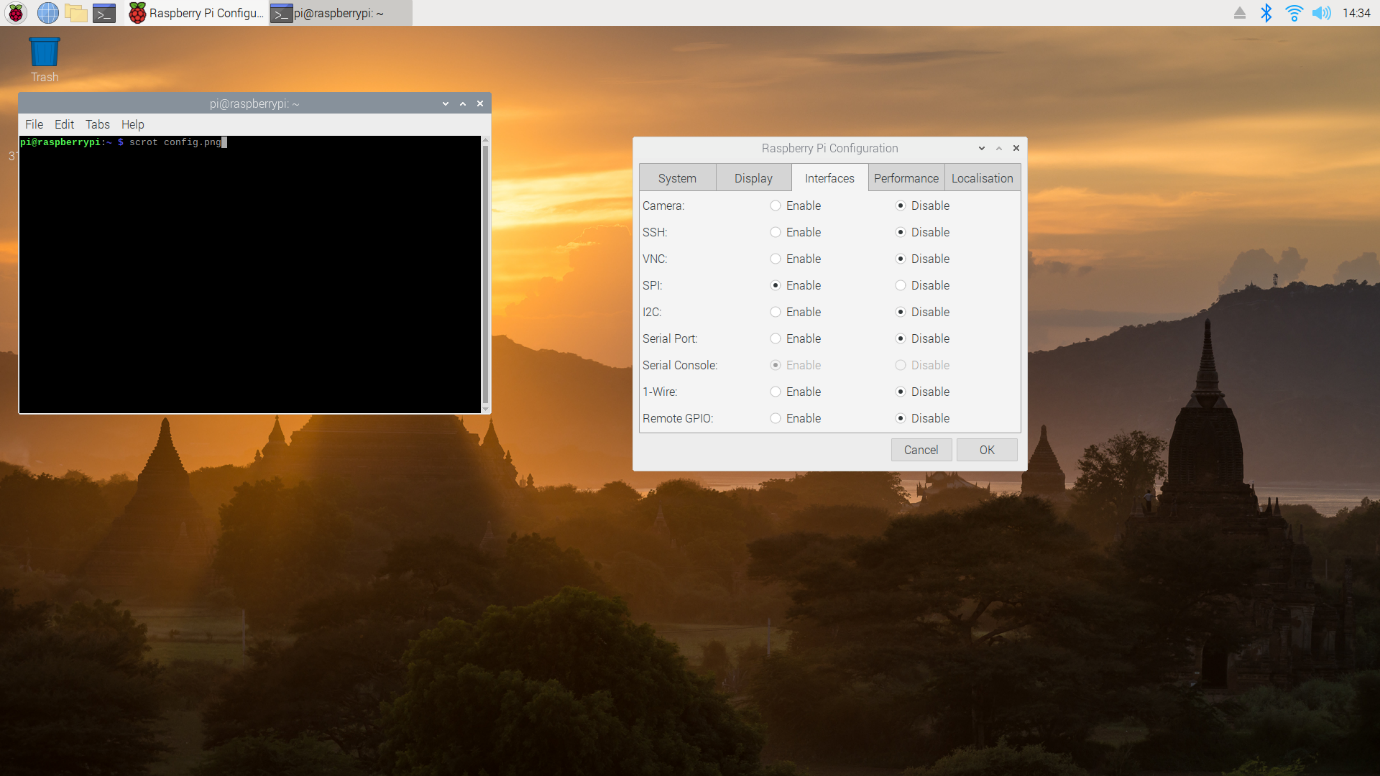
Raspberry Pi Physical GPIO Header:



**Instructions – Code**

**Setup**

Make sure that the SPI interface is enabled. Run the “Raspberry Pi Configuration” app, which can be found under the Raspberry Pi Start button, select “Preferences”, then “Raspberry Pi Configuration”. When the app runs, select the “Interfaces” tab and check that “Enabled” is selected next to SPI (if not, click it now!).



Create a folder for your project, in the following example it is named SmartWatch.

Copy the “lib” and “Font” folders into this project folder.

I suggest that you use “Thonny” as your development IDE (at least initially). You can find this under the Raspberry Pi launch button in the Programming group.

Create a folder called “example”, run Thonny and create the “SimpleTest.py” program in this folder (see below). Press the big green button to run, and you should see the message “Hello World” appear on the Smart Watch display.

If you do not see the message:

1. Check for any error messages in Thonny
2. Ensure that you have enabled the SPI interface
3. Double-check your connections
4. Ask Mr Gamble

**Example**

The following program simply renders a message on the Smart Watch display. The program imports a library (“LCD\_1inch28.py”) to interface with the Smart Watch display, and we assume that this library is stored in a folder called “lib” which is located in the immediate parent folder (“..”).

The program uses Cairo, which is a library for drawing vector graphics. This allows us to create images programmatically, and then to send them to the Smart Watch to be displayed.

import sys

sys.path.append("..")

from lib import LCD\_1inch28

from PIL import Image

import cairo

try:

disp = LCD\_1inch28.LCD\_1inch28()

# Initialize library.

disp.Init()

# Clear display.

disp.clear()

# Create a new drawing surface

surface = cairo.ImageSurface(cairo.FORMAT\_RGB24, 240, 240)

ctx = cairo.Context(surface)

# Write "Hello World" in the centre of the drawing surface

ctx.set\_source\_rgb(1, 1, 1)

ctx.set\_font\_size(30.0)

ctx.select\_font\_face("Quicksand",

cairo.FONT\_SLANT\_NORMAL,

cairo.FONT\_WEIGHT\_NORMAL)

disp\_text = "Hello World"

(x, y, width, height, dx, dy) = ctx.text\_extents(disp\_text)

ctx.move\_to(120 - width/2, 120)

ctx.show\_text(disp\_text)

# Update image on display

disp.ShowImage(Image.frombuffer('RGBA', (240, 240), surface.get\_data(), 'raw', 'RGBA', 0, 1))

except IOError as e:

print(e)

I strongly suggest that you set up a GitHub repository for code and documentation – this will make it easier for you to share code.

**Additional Information**

**Serial Peripheral Interface (SPI)**

The Serial Peripheral Interface (SPI) standard provides a simple way to connect a “Controller” (such as the Raspberry Pico) to a range of devices (“Peripherals”) including sensors, memory cards, and displays, such as the LCD display used in this project.

The main signal lines are:

* COPI (Controller-Out, Peripheral-In)
* CIPO (Controller-In, Peripheral Out)
* CS (Chip Select)
* SCK (Clock output from Controller)

For the LCD display, COPI is labelled DIN; CS is labelled CS, and SCK is labelled CLK. Other display connections are device specific, such as the BL (backlight).

In this project we are only using one-way (controller-to-peripheral) data transmission, but SPI supports two way transmission. SPI can support multiple peripherals connected to a single controller, but again in this project we are just communicating with the display.

To find out more about SPI, try these links:

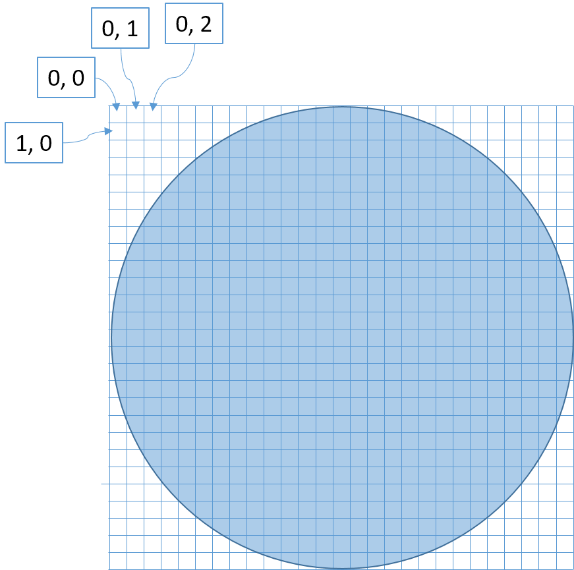
<https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all>

<https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html>

(Note that the second link uses older terminology of ‘master’ and ‘slave’ rather than ‘controller’ and ‘peripheral’.)

**Waveshare 1.28” LCD Module**

The display uses an inscribed circle model for addressing:



Note that this means that some pixels are outside the valid display. You can write to these pixels, but nothing will appear on the display.