Internet-connected displays make the Industrial IoT more visible

By Mark Patrick, Mouser Electronics

This article explains the easy development of full-colour displays for Industrial IoT environments. Prototyping of ideas, as well as developing production code, is made simple through clever integration into the Arduino IDE. And, via the integrated WiFi transceiver and software library, Industrial IoT applications are easily incorporated into cloud services.

As the implementation of the Industrial Internet of Things (IIoT) starts to take place, the whole concept is starting to become a little more tangible. Manufacturing equipment is becoming network connected and even being attached to the cloud, with innovative mobile apps putting equipment status in the palms of our hands. Despite this, not all manufacturing environments are conducive to carrying around a portable device, even if ruggedised. Additionally, not all employees within an organisation require an expensive resource, such as a tablet, just to display the status of the equipment they are working with.

Most new machines will have their human-machine interface (HMI) integrated into their housing. However, retrofitting of old machinery or customisation of new machines will continue to be a common approach in the move to more IIoT-enabled equipment in the coming years. In harsh working environments, where liquids, dust and humidity require that enclosures have appropriate IP ratings, the addition of yet another hole for a display is understandably called into question. Having balanced the costs and potential for quality issues against the benefits, the display often falls off the feature list in favour of solutions with a simpler sealing. Additionally, the mounting location of a grey control box is not always optimal for the machine operator.

With the introduction of their wireless Internet of Displays product range, 4D Systems tackles all these issues with a family of colour displays that are wireless, easy to program, and simple to integrate. If the information can be accessed through a wireless network at the site of operation, it can now be prominently displayed in a location where an operator works, and a power source is available. The IOD-09 is one such product (figure 1). Available in both through-hole and surface-mount options, it integrates an 80x160-pixel colour display, 802.11 b/g/n/e/i WiFi transceiver with ceramic chip antenna, TCP/IP protocol stack and 4Mbit (512KB) of flash memory. The entire device is powered by the Espressif ESP8266 SoC that provides 128KB SRAM, 80KB of which are available for the user application. The entire solution is neatly integrated into a black metal bezel of 31.8x16.4mm and weighs just 5g. To round off the feature set, a microSD card pull/push type socket supports media of 4GB and above, providing space for media files and a location for storing data logs.

The IOD-09 is fully programmable and can be used standalone to interface with web-based services to display text, images, icons or widgets on its display. It offers 6 general-purpose input/output (GPIO) pins that can also function as alternate functions including I2C, SPI and 1-Wire interfaces and a single PWM capable output. A TTL level asynchronous serial port provides all the expected standard features with Baud rates from 300 to 921600 supported. In the event that a more demanding product is planned, the device, via these interfaces, can be easily interfaced with a low-cost 8-bit microcontroller, allowing inputs via buttons, capacitive touch or joysticks, and the addition of LED indicators and a sounder for audible feedback. Perhaps one of the most impressive elements of the entire package is the ease with which it can be programmed, as it slots neatly into the Arduino IDE programming environment after a little initial configuration. With more and more developers turning to Arduino, the simple setup-and-loop approach to application development, along with the well documented API of the libraries, makes code a breeze to create.

As the ESP8266 SoC at the heart of the IOD-09 solution is not supported within the default Arduino IDE configuration, there are a couple of steps to be taken before software development can begin in earnest. These are adequately explained in the datasheet. The process starts by downloading and installing the standard Arduino IDE as explained in chapter 11.1 of the datasheet. The collection of build tools and configuration information that the Arduino IDE needs to build a sketch is added to the environment by adding a URL

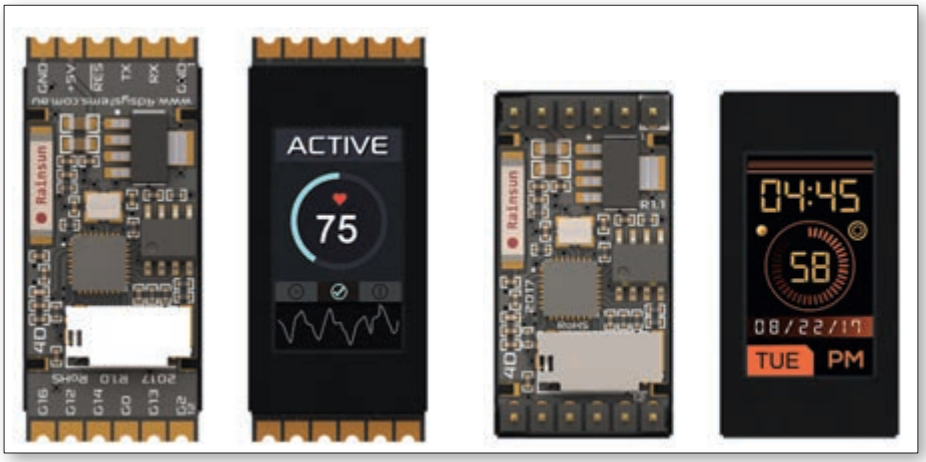

Figure 1. IOD-09 - images of surface-mount and through-hole versions

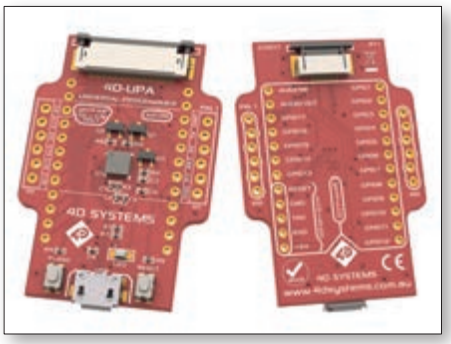

Figure 2. 4D-UPA programmer

to the Additional Boards Manager URLs: field in the preferences window. The next step involves closing the Arduino IDE in order that one of the files downloaded (boards.txt) can be modified by hand. Again, this is adequately explained in the instructions, with the full path to the file boards.txt provided for all operating systems. This file is used by the Arduino IDE to understand, amongst other things, which tools should be used to build the code, which clock frequency the processor uses, and how much memory the platform has. Additionally, it forms some of the options that appear under the Tools menu item in the IDE when a specific board has been selected.

During our evaluation, the IOD-09TH was being used. In order to program the device, 4D Systems provides the 4D-UPA Programming Adapter (figure 2), available separately, to which the IOD-09TH, with its through-hole 2.54mm pitch pin mounting, can be easily fitted. The 4D-UPA connects to the developer computer via a micro USB cable. Following the information for this device on from the 4D Systems datasheet we were able to download the appropriate driver (URL 4). Once installed, it was possible to download our demonstration sketch to the target and tick-off our first success. To trial the capabilities of the display, a simple application was developed that might be found in an industrial environment. The assumption was made that a machine had been upgraded to connect it to an IIoT environment and required a small display to show an oil pressure dial and the number of hours of operation (figure 3). In addition, when a maintenance schedule point is reached, the display should show that the maintenance engineer has been scheduled to visit (figure 4). The operator is then able to double check that the maintenance has been conducted, as the message will be cleared via the IIoT cloud app upon completion.

Figure 3. Demo displaying PSI dial and operational hours

Figure 4. Demo additionally displaying that the machine's maintenance engineer has been scheduled to visit

In order to access the graphics library, a simple #include GFX4dIoD9.h is all that is required. The IOD-09 display is initialised in the set-up()-function of the Arduino sketch. An example for this is provided in the Arduino Libraries Reference Manual (chapter 1). Like most displays, the individual pixels of the display are not square but rectangular. Thus, to create a true circle for the oil pressure dial, it is necessary to draw an ellipse using gfx.Ellipse(). In total, three calls to this function were used to draw a circle of three pixels in width. The dial pointer also needs to be created. To provide a simple 3D effect, four triangles in two colours were drawn, with the opposing triangles in lighter and darker blue. In the demo, the pointer angle is fixed and requires a little mathematics to calculate the angle and draw the triangles appropriately around the centre point. These filled triangles were created using gfx.TriangleFilled().

The IOD-09 also provides integrated support for displaying text in its library. Using gfx.MoveTo() the starting point for each line of text can be defined. By default, text that extends beyond the edge of the display wraps around. Output of the text is made possible with gfx.print(). To handle the warning regarding impending maintenance, the text colour is changed to red. There are a couple of points to note here. Firstly, changes to colour are permanent. Therefore, it is necessary to return the colour to its original value once drawing the Maintenance Scheduled text is complete. If this is not performed, all further text will be draw in red. Secondly, it is important to define a background and foreground colour for text. If this is forgotten, the text will be written over the previous text, causing a mess of characters on top of one another. The function for this is gfx.TextColor(<FOREGROUND>, <BACKGROUND>) and is shown, along with the core of the rest of the sketch code in figure 5. It is also worthwhile examining the content of the GFX4dIoD9.h header file as a huge range of potential colours are already predefined.

Coding a gauge and indicator by hand does require some planning, and squared paper is useful whilst determining the position of each pixel. 4D Systems also provide a full development environment named 4D Systems

Figure 5. Demo code screenshot

Workshop4 IDE for those who prefer faster development and gauges and dials with a more professional look. The IDE generates code that can be quickly integrated into an Arduino sketch and downloaded onto the IOD-09. The libraries included with the IOD-09 also include everything necessary for access to the Internet via the integrated WiFi function. The demonstration created here uses fixed values for the display output, but connection to a secured WiFi access point is easily achieved in around 15 lines of code. From there it is only a small step to accessing a JSON encoded file from a server or making use of an IoT cloud service that can share the data required.

Embedded development engineers will already know that the inclusion of a full-colour display of any dimension requires a significant amount of effort. Selection of a display, finding a suitable display driver, acquiring a microcontroller with enough performance and locating a graphic library are all huge undertakings. The 4D Systems Internet of Displays solutions is, therefore, a welcome relief, enabling product developers to deliver a great-looking display for their product without having to deal with the associated development costs. Prototyping of ideas, as well as developing production code, is made simple through the clever integration into the Arduino IDE. And, via the integrated WiFi transceiver and software library, IIoT applications are easily incorporated into cloud services. ■

Visit Mouser at electronica, Stand C3-550. Experience Augmented Reality, discover layers of infrastructures supporting smart cities, and spin our wheel of fortune for a chance to win a dev kit!