A PROTOTYPE TABLET COMPUTER

Abstract

Today because of the progress in both hardware and software the difficulty of creating a personal portable computer or a table has reduced a lot from the perspective of cost and hardware knowledge. What I am trying to accomplish with my project is to demonstrate the steps it takes to build a tablet, and more exactly how to build a simple operating system for it. The main focus of this project is building an operating system for an Arduino which will only have as an input output device a touch screen. Even though the software is more important for my project in this literature review I would also like to show the relevance of hardware.

Andrei Brabete

I certify that all material in this dissertation which is not my own work has been identified.

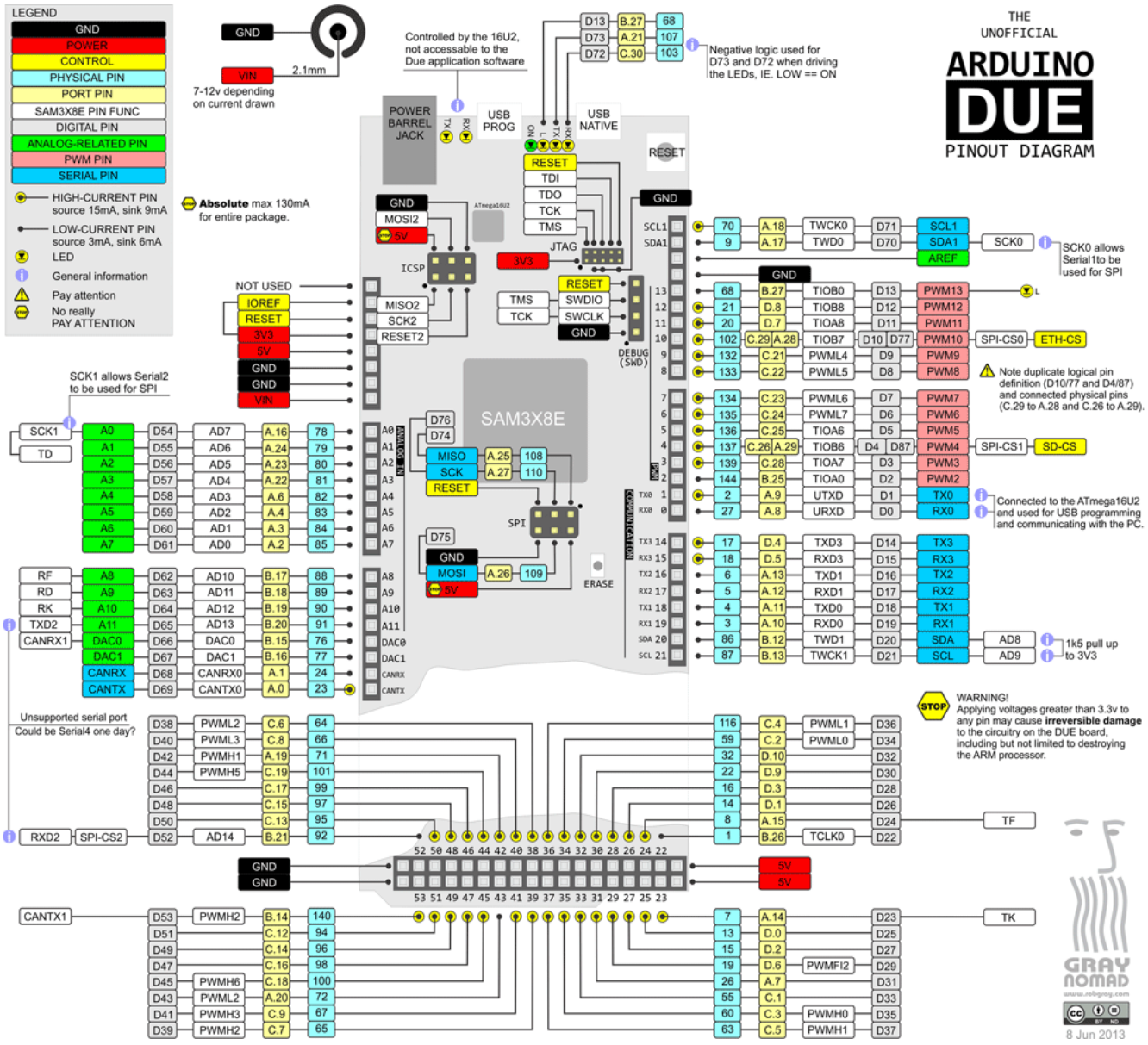
The project that I am working on is to build a prototype tablet based on an Arduino embedded computer. If I want this project at least to resemble a tablet, it will need to have a way of controlling all the processes and manage the memory, in other words it will require an operating system. The operating system will be written in C, because of the many advantages it has to bring for this specific project, like, C is already the programming language for Arduino and is the one of the fastest programming languages, and also offers a great control over the memory.

Taking a look over this project from a higher level, it can easily observe that there are two main areas to be looked into them and dealt with. Firstly is the hardware part of the project, and secondly the software aspect, or the operating system aspect. Till now the main focus was on the hardware part of the project, researching and selecting the proper parts to build the tabled. I would like emphasise that, even though at a first glance the hardware selection is not a complex task, making the wright choices right now may absolve from future problems and overcoming on the deadline. The first choice was selecting the Arduino controller. Nowadays there are several models of Arduinos, so I went for the one that was the best for my project, Arduino Due. I selected this specific model of Arduino because of the increased processing power and memory available compared to the other models available and also of the small size of the board. From the description available on the official Arduino website [1] we can see the main features of this board: "The Arduino Due is a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU. It is the first Arduino board based on a 32-bit ARM core microcontroller. It has 54 digital input/output pins (of which 12 can be used as PWM outputs), 12 analog inputs, 4 UARTs (hardware serial ports), a 84 MHz clock, an USB OTG capable connection, 2 DAC (digital to analog), 2 TWI, a power jack, an SPI header, a JTAG header, a reset button and an erase button.". Beside this features, the other aspects that are relevant for my project are the size of the flash memory, 512KB all available for the user, and the size of the SRAM, 96KB. At the moment the board is powered by an AC-to-DC adapter at 9V, but in the future, when the table will be stable, there is the option to change it with a battery, offering a better portability.

The other hardware component of my project that is important is the touch screen which will act as both input and output device. Here the search and selection was harder, compared with the Arduino board. Mainly because of the large number of touch screens available on the market and the small number of already existing projects that used a touch screen with an Arduino. I started my search for the right hardware by firstly analysing the existing Arduino computer tables, and shortly I discovered that there were only a handful of them built. Another key aspect important for my project is to find a screen with specific libraries for the Arduino. This will help me advance faster with the graphical interface, and not starting and trying to write my own code for drawing a line. The project that I am working on is centred on creating an operating system that will be able to have a decent memory

Figure 1

http://www.robgray.com/temp/Due-pinout-WEB.png



management system and a processor time scheduler, and removing the basics aspects of graphics, having to deal with them only at a higher level, I am able to concentrate more on the core of the operating system. The first working project that was similar to what I am trying to build was made by a company called Vizic Technologies. The have developed a series of touch screens devices specific for the Arduino boards. The only drawback of these displays were the small size of them, 1.8", 2.4", and 3.5" inches. They have a bigger screen of 4.3" inches, but for this to work with the Arduino board, it requires a shield, to make the connection between them possible, because the screen requires more current that what the Arduino can provide from the 3.3V output pin [2]. A key feature of this specific 4.3" inch display is shown on the official web page: "The SmartGPU 2 is a powerful easy to use, intellectual property, graphics, audio, touchscreen and full data logger processor embedded in a state-of-the-art ARM Cortex-M3 chip; this is mounted on a board with a touchscreen colour LCD. It's aimed to help developers to create advanced Graphical User Interfaces (GUIs) in a very easy way, the Smart GPU 2 processor doesn't need any configuration or programming on itself, it's a slave device that only receives orders, reducing and facilitating dramatically the code size, complexity and processing load in the master host processor. It features high end FAT32 format data management functions (Data Logger) to create even more advanced applications in just minutes, not days."

The more I searched for bigger screens, the more I found out that, even though there are many 7" inch screens, none of them had a library for the Arduino, and only a small number of them had an embedded processor on them, to act as a slave device and ease the work load of the Arduino processor, which frankly in my opinion will be too much for the Arduino to also have to compute the graphics. For the displays that I found which were 7" inches and had an embedded processor on them, there was no evidence of them working with the Arduino as a controller.

Once I starting looking again at 4.3" inches screens I discovered one which had an embedded processor on and libraries special developed for the Arduino boards, also with evidence of it working along with the Arduino. The touch screen that I selected for my project is created by a company from Australia, 4D Systems. They develop a wide range of display for a diverse use, from screens made for the Arduino to ones made for Raspberry PI. The display that has optimal features for my project is the μLCD-34PT unit from 4D Systems, which is a resistive touch screen, and it can be found on their official web page [3]. The description from 4D Systems web site highlights some of the key aspects of this unit: "The uLCD-43 is an intelligent graphics display that harnesses the power to deliver a diverse range of features in a single, compact cost effective unit. Embedded at the heart of the design is the PICASO processor, which is driven by a highly optimised virtual core engine; EVE (Extensible Virtual Engine)." One of the aspects that made this display a better choice for me over the one from Vizic Technologies is that I do not need to use a shield to connect the screen to the Arduino. They also offer a Shield as an option, but I do not think it will bring any major advantages for the project that I am trying to make. One of the feature that the shield brings is a rapid way to

reset the screen and make sure that is not out of sync with the Arduino, but this can be dealt with from the code, which I will explain later on. The shield makes both the Arduino and the screen more stable in the case when there are other modules added to the Arduino, but for me this is not the case, the screen being my only module added to the Arduino, this being the case there was no good cause to have a shield, which only will make the entire unit heavier and bigger. The screen has a resolution of 480 x 270 pixels and 65k colours, and the voltage range operation is between 4V to 5.5V, which is perfect for the 5V output pin from the Arduino. The communication between these two devices will be made by 5 jumper cables, and it will be a serial communication with the bound between 300 and 256K, and the display will act as a slave for the Arduino.

The 4D Systems screen unit also has a handy tool to ease the difficulty of the graphical interface design, which is their IDE, Workshop4. The IDE, Workshop4, is composed of four main development environments to build the project you are working on. The first one is called Designer: " This environment enables the user to write 4DGL code in its natural form to program the display module." , the second one is ViSi: " This environment is also provided to transform the display module into a slave serial display module, allowing the user to control the display from any host microcontroller or device with a serial port." , the third one is ViSi – Genie: " An advanced environment that doesn't require any 4DGL coding at all, it is all done automatically for you. Simply lay the display out with the objects you want (similar to ViSi), set the events to drive them and the code is written for you automatically. ViSi-Genie provides the latest rapid development experience from 4D Systems." And the last one is Serial:" This environment is also provided to transform the display module into a slave serial display module, allowing the user to control the display from any host microcontroller or device with a serial port." [4]. For my project I will use the ViSi – Genie environment, to ease the difficulty of managing the graphics and create an appealing interface to the user which will have to follow some already define norms. Next I will describe how the connection between the Arduino and the 4D Systems touch screen unit is made and explain how it works. The connection, how I said before will be made by a 5 way cable, coming from one of the two programming headers that are found on the touch screen unit, this is the default configuration, which can be modified from the Workshop4 IDE to use the expansion header from the 4D Systems unit, but for now I will be using the default one. The

Connection Using Jumper Wires

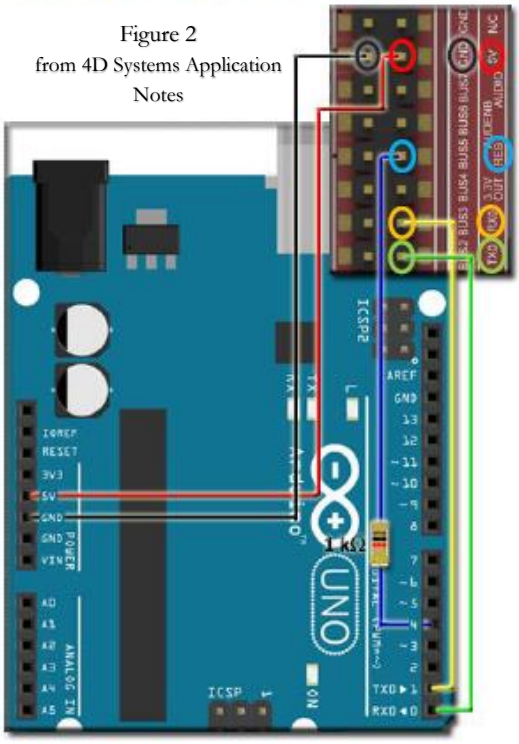


Figure 2
from 4D Systems Application Notes

five cables have their own function as follows: the first wire will be the +5V one which will supply the display with power from the Arduinos 5V pin which can be seen in Figure 1, the second one is the Rx wire (receiving) going into the Arduino RX0 pin which can be seen in the schematic of Figure 1 as physical pin 27, the third cable is Tx (transmission), which will go the Arduino TX0 pin, physical pin 2, the fourth wire is the GND (ground) which can go to any of the ground pins from the Arduino, the last wire is the RES (reset) one and this will to the Arduinos physical pin 137, later on I will explain the reason to have this wire. In Figure 2 you can see an example of connection between an Arduino Uno and the expansion header of a 4D Systems display unit. There is no difference between the pins used on the Arduino Uno and the ones I will be using for the Arduino Due. I will use for the moment the default programming header from the touch screen unit. Also because the Arduino Due has multiple Tx and Rx pins I can use a second connection between the two units to have a better efficiency, but for now I am pleased to use only one connection, when the time for adjustments and optimisation will come, I can easily configure the second connection.

To return on the display unit part, the graphics project constructed with the Workshop4 IDE will be loaded onto a microSD card and mounted to the screen, the card will held all the information that the screen unit needs to draw the corresponding objects on the screen and store any other images and audio files needed by the project to run properly.

The next step is to show how to initialise the actual communication between the two units from the software perspective. There are some steps that must be followed to be able to control the screen from the Arduino. The first step is to get the specific build library for Arduino from GitHub [5], and add it to the Arduino libraries, after this is done you must include this library into your program to be able to control the display. The next thing to do is open a serial port to let the Arduino communicate with the scree, as showed in Figure 3, which as Figure 2 are both from a document created by 4D Systems to help understand and work with their products [6]. The default baud used for data transmission as seen next in Figure 3 is 200000, which is fine for now and I will leave it like this. The next step is making sure that the display is ready to receive commands from the Arduino, and this is done through the RES wire, which needs to be initialised and send the display an impulse, preparing it, this will make sure that the Arduino and the screen are not out of sync and not interpreting the tasks from Arduino in a wrong way. To define this pin you need to use the following command: *#define RESTLINE 4*, this refers to the Digital pin 4 from the Arduino, physical pin 137. Now the following step is to reset the display, and in my case, using jumper cables need to add a few lines to do this, as seen in Figure 4.

Open a Serial Port and Set the Baud Rate Figure 3

In the **setup ()**, the line shown below sets the data rate in bits per second (baud) for serial data transmission of the port Serial0.

```
Serial.begin(200000);  // Serial0 @ 200000 (200
```

The next line indicates that the ViSi-Genie-Arduino library will use the port Serial0 for communicating with the display.

```
genie.Begin(Serial);   // Use Serial0 for talki
```

Figure 4

```
pinMode(RESETLINE, OUTPUT);  // Set D4 on Arduin
digitalWrite(RESETLINE, 0);  // Reset the Displa
delay(100);
digitalWrite(RESETLINE, 1);  // unReset the Disp
```

At this point the screen is ready to take commands from the Arduino. This specific screen that I am using the uLCD-43PT, it can set the contrast of the screen from the values 0 to 15, 0 meaning the screen is off, and at 15 the screen is set at the maximum contrast. Is recommended to set the screen contrast before using the screen to make sure the screen is on, this is make with the following line: "*genie.WriteContrast(1);*", or a bigger value depends on the particular configuration that you are looking for. All the information and figures about using the display with the Arduino is taken from the Application Notes document provided by 4D Systems.

In the example code provided by the 4D Systems it demonstrate how to operate the display and how to use the library provided, is the first step to do before trying to create your own application. It illustrate the foundation that a project needs to work properly and brings insight on what is happening from the code point of view. The example code uses a few of the graphics objects provided by the ViSi – Genie development environment like: LED digits, Slider, Static text label, User LED, Gauge and Strings, this can be seen in the Figure below.

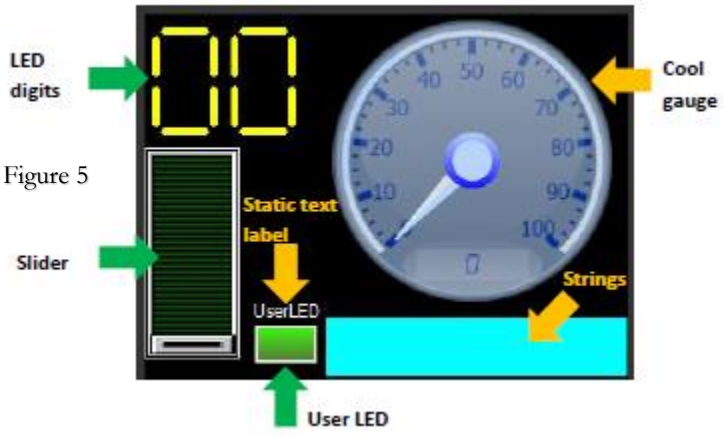


Figure 5

The LED digits will show the value of the Slider, and this computational part is done by the Arduino, with the help of Event Handlers. The event will be invoked when some input is received by the touch screen and sends the appropriate information to the Arduino controller, which will respond as programmed. The value of the Gauge is set by the Arduino. What I am trying to emphasise with this example is

that the touch screen display acts as a slave and also that a lot of the work load is done by the embedded processor found on the display unit.

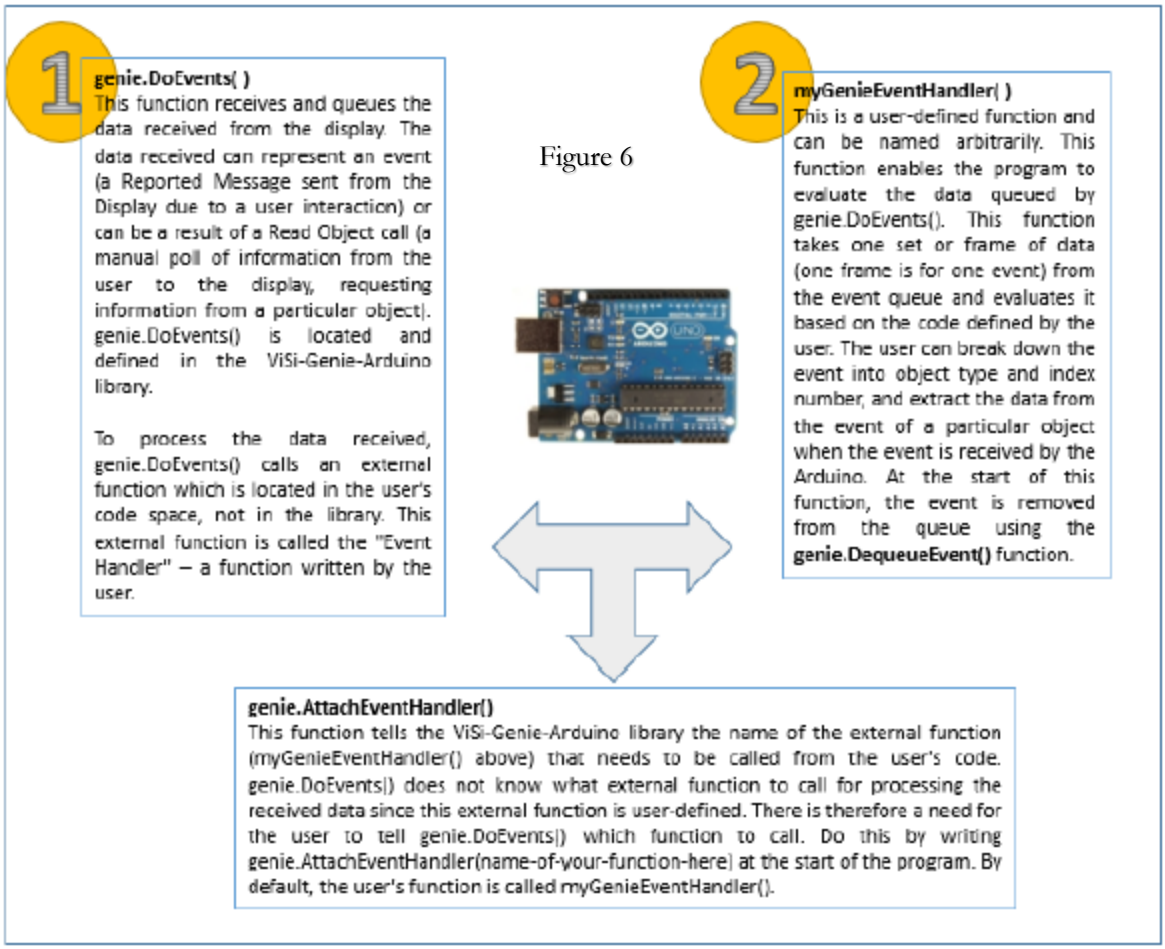


Figure 6

In Figure 6, above, the flow of commands and the logic behind the code for the ViSi – Genie development environment is easily described and explaining the role of each part. At this point I made very clear the understanding of the hardware components in my project and how to use the advantages offered by them to create a more robust system. Each step in choosing the right component will have a bigger impact later on, when I will start working on more complex parts of the operating system. The fact that I wanted to cover till this point was the importance of the right hardware for the project that someone is working, not only for me, but exemplified with my specific project, the need of a solid foundation to help you construct and develop what you have in mind without any later surprises.

From my point of view, to have better odds to come up with a working tablet, I need to take an Agile approach, and that is why I started selecting the hardware and have a few experiments with it to have an understanding of the configuration. Instead of creating a design for the entire operating system and after that getting the hardware and start implementing the design.

As long as it goes for the software part of the project I start looking into Free RTOS, which is one of the most known real-time operating system kernel. This system was design to be small and efficient, having only around four C files, with some assembly functions to manage the processor register to enable a sort of multi-tasking. The kernel of this operating system is a microkernel and been design to operate on multiple platforms, including the ARM, which can be also find on the Arduino Due board. I think this kind of design will be optimal for my project, being quite a small operating system and also having the facility of switching between tasks. Because of this features I will try and create my operating system on the principles of Free RTOS. Also the documentation provided by Free RTOS on their official website is very clear and explaining the implementation of the system [7]. One of the configuration of the FreeRTOS for the scheduler is pre-emptive, which is a good design choice for my objective. The next steps I will do for my project will include mainly the software part, hopefully if everting will be fine on the hardware part, is doing more researching into FreeRTOS and into well-known algorithms for operating system memory management and processor scheduling, a book that is one of the best for this kind of work is Operating System Concepts [8] . I will not try to create new algorithms or way to create an operating system my aim is to find out the perfect combination and design for the hardware and purpose of my tablet.

Bibliography

[1] Arduino, “Arduino Due,” Arduino, 2015. [Online]. Available: https://www.arduino.cc/en/Main/arduinoBoardDue. [Accessed 5 Octomber 2015].

[2] Vizic Technologies , “SmartGPU 2 - LCD480X272 - 4.3",” Vizic Technologies , 2015. [Online]. Available: http://www.vizictechnologies.com/#!smartgpu2-lcd480x272/c18hh. [Accessed 29 September 2015].

[3] 4D Systems, “uLCD-43,” 4D Systems, 2014. [Online]. Available: http://www.4dsystems.com.au/product/uLCD_43/. [Accessed 06 October 2015].

[4] 4D Systems, “Workshop4 IDE,” 4D Systems, March 2015. [Online]. Available: http://www.4dsystems.com.au/product/4D_Workshop_4_IDE/. [Accessed 10 October 2015].

[5] 4D Systems, “ViSi-Genie-Arduino-Library,” GitHub, 08 October 2015. [Online]. Available: https://github.com/4dsystems/ViSi-Genie-Arduino-Library. [Accessed 20 October 2015].

[6] 4D Systems, “ViSi-Genie Connecting a 4D Display to an Arduino Host,” 13 August 2015. [Online]. Available: http://www.4dsystems.com.au/downloads/Application-Notes/4D-AN-00017_R_1_01.pdf. [Accessed 15 October 2015].

[7] Real Time Engineers ltd., “FreeRTOS,” 2010. [Online]. Available: http://www.freertos.org/index.html. [Accessed 20 October 2015].

[8] P. G. a. G. G. A. Silberschatz, Operating system concepts, New York: John Wiley & Sons, 2002.