Final Project

3D Printing Machine

Maxim Goukhshtein, 260429739

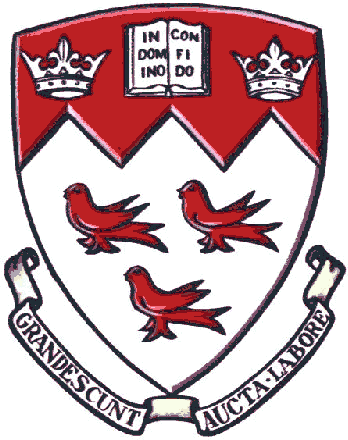
Olivier Laforest, 260469066

Nuri Ege Kozan, 260359680

Geneviève Nantel, 260481768

Department of Electrical and Computer Engineering

McGill University, Montreal



April 21st, 2015

Group 4

ECSE 426 Microprocessor Systems

Presented to Professor

Zeljko Zilic

# Abstract

The goal of the final project was to create a 3D printing machine which is controlled wirelessly by a command board. Using the keypad, a user selects the desired shape of his/her choice and the graphical LCD board will display the chosen shape on the screen. Subsequently, the plotting/printing commands are sent wirelessly to the controller board which controls the motors in order to draw the desired shapes on the whiteboard. The x-y-z coordinates representing the shapes are sent wirelessly to the 3D printing controller board, and then, are subjected to various calculations in order to convert these coordinates into motor angles. The whiteboard marker is held by two 3D printed plastic arms and it is positioned by two stepper-motors which place the arms at the desired x-y location. The remaining stepper-motor is used to lift the structure containing the two locating motors up and down. Three simple pre-defined shapes consisting in a square, a rectangle and a triangle can be printed on the whiteboard as well as “on-the-fly” shapes. The “on-the-fly” settings are preconfigured on the keypad and map to certain directions the user wants to draw. Finally, in order to successfully realize this project, the various tasks are implemented across multiple threads using the RTX RTOS, helping to initialize, synchronize and maintain the proper functioning of the different components. This report will go into the details of the theory, implementation and testing of the 3D printing system and its various components.

# Problem Statement

The goal of this project is to design and implement a 3D printing machine controlled wirelessly. The overall system needs to be composed of two subsystems, the transmitting system and the receiving system. The transmitting system is composed of the STM32F429 Discovery board, a keypad and the TI CC2500 wireless chip, and should allow the user to input its desired commands, record and process them, and send those commands wirelessly to the receiving system. The receiving system needs to be composed of a STM32F407 Discovery board, a TI CC2500 wireless chip, three HiTech HS422 stepper motors and the 3D printing machine made out of the six 3D printed plastic parts. The receiving system needs to be able to receive the user selected command, sent from the transmitting system, and transform those commands into motor commands which are going to set the 3D printed parts in motion to realize the appropriate drawing. Three separate pulse width modulated (PMW) signals should be used to control the stepper motors. The PMW signals must be generated through hardware timers.

The overall system needs to implement the following two modes of operation:

* **Predefined shapes mode:** In this mode, the user can select one of the three predefined shapes, a square, a rectangle or a triangle, and once selected, the plotting commands are sent and the desired shape is printed. Each predefined shape should be mapped to a keypad button and a fourth button should allow the user to send the plotting command wirelessly.
* **On the fly drawing mode:** In this mode, the user can have the printer draw successive straight line segments, of lengths equal to , , , etc. The user can select the orientation of each new segment through a preselected set of keypad buttons. The first segment should start at the coordinate, which should be located in the lower left corner of the printing area. Four different buttons should allow the user to orient the next segment to be drawn up, down, left and right, which correspond to , , , respectively, assuming that orientation is the horizontal positive axis as per the Cartesian coordinates. Another set of four buttons should allow the user to increment the orientation angle in steps of, starting from , decrement the orientation angle in steps of starting from , increment the orientation angle in steps of starting from , and decrement the orientation angle in steps of starting from , respectively. One of the keypad buttons should allow the user to reset the drawing back to location . Finally, a button should allow the user to send the current straight line segment to be drawn by the 3D printer. Each subsequent segment is drawn starting from the end of the previous one, unless the reset button is pressed. The LCD display of the STM32F429 board should display all the segments which have been drawn so far, and the latest segment should be blinking on the screen.

In addition to the two mode previously described, either the MEMS accelerometer sensor of the STM32F407 board of the receiving wireless system, or the gyroscope of the STM32F429 board of the transmitting wireless system should be used to allow the user to control the position of the printer with respect to the roll and pitch angles of the selected board. In this way, the user can dynamically control the 3D printer in real time. An additional extra feature also needs to be implemented. The choice of this extra feature is left to the implementers and should be creative and technically challenging. Finally, the LCD screen on the STM32F429 board should be used to provide visual feedback to the user (i.e. mode selected, shape to be drawn, etc.).

# Theory

In order to move the motors to the appropriate x and y values (i.e. the x and y coordinate of the pen) sent by the transmitter board, calculations need to be done in order for the motors to move to the appropriate angles. The calculations below convert x and y values to angle values that will be sent to change the pulse width modulated signal to the motors. Our system, shown in Figure 2, can be represented by a pentagon as shown in Figure 1.

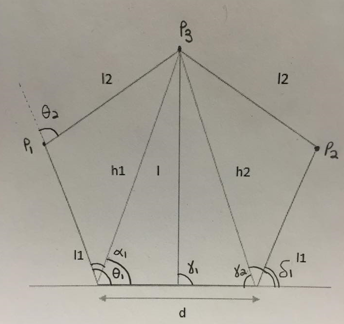
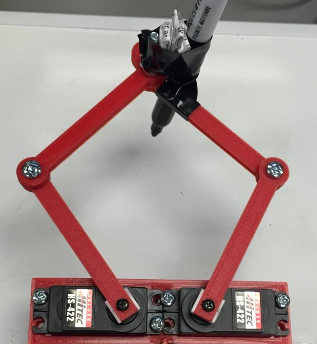
 

Figure 1: Pentagon Theoretical Representation  Figure 2: Pentagon Hardware Representation

The following cosine and sine mathematical laws were used in order to do the calculations for properly converting the x-y values to motor angles. The first step is to calculate the length which is the length from the marker to the position exactly between the two motor pivots:

|  |  |
| --- | --- |
|  |  |

With this, can be calculated using the cosine law stated as shown below.

|  |  |
| --- | --- |
|  |  |

Similarly, can be computed with

|  |  |
| --- | --- |
|  |  |

Using the sine law mentioned below, can be calculated with

|  |  |
| --- | --- |
|  |  |
|  |  |

Similarly, can be calculated using the sine law

|  |  |
| --- | --- |
|  |  |
|  |  |

Using the cosine law again, and can be calculated with

|  |  |
| --- | --- |
|  |  |
|  |  |

And finally,

|  |  |
| --- | --- |
|  |  |
|  |  |

and are the values that are used to control the duty-cycle for the PWM of the motors.

# Implementation

## Motor Configuration

The motors are controlled using a hardware pulse-width modulation (PWM). To do so, a hardware timer that supports PWM must be chosen as well as the channels on the timer. TIM3 was used as it supports PWM. The table shown in Figure 3 shows all possible pins for the TIM3 timer and each channel. Each output channel can only support a maximum of 3 pins.

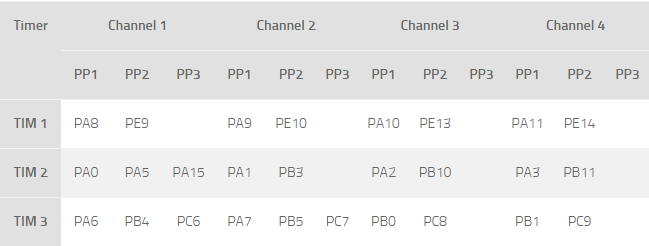


Figure 3: TIM3 PWM Channels and Pins [1]

The prescaler and period were then set. The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx\_PSC register). Basically, it takes a basic timer clock frequency and divides it by some value. It allows the timer to be clocked according to user’s choice. The desired rate is defined as follows:

|  |  |
| --- | --- |
|  |  |

Furthermore, the Output Compare (OC) mode and PWM were subsequently configured. OC is used to measure duty cycle, period and frequency, and PWM is used to produce a square wave with a particular duty cycle, period, and frequency. In other words, OC is the input and PWM is the output. The OC function is used to control an output waveform or indicating when a period of time has elapsed. When a match is found between the capture/compare register and the counter, the output compare functions. The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing 110 (PWM mode 1) or 111 (PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register. PWM2 clear on compare was used. Finally, each motor was associated to one pin and each pin was associated to an OC channel where the pulse was adjusted in accordance to the incoming angle.

## Keypad Controller



Figure 4: Keypad used in the experiment

The instructions received from the keypad, shown in Figure 4, in the transmitter board are then translated into simple drawing commands, which are wirelessly sent to the receiver board. The initialization and the structure of the keypad was set up in the same way as in the previous labs, with the addition of extra de-bouncing features that would only read the pressed button once, even if the button was kept down. The specific commands used for the instructions were:

* Buttons A, B and C were used to switch between different modes of operation. Mode A was used for the predefined shapes, Mode B was used for the on-the-fly and MEMS mode, and finally mode C was used to draw numbers, which is the additional extra feature.
* Pound button (#) was used to reset the position of the 3D printer to the initial position at coordinate (0, 0).
* Button D was used as the "enter" button, which is required to be pressed every time after the user selects their desired instruction to take effect. When this button was pressed, the desired command was wirelessly transmitted to the receiver as well.
* All of the other remaining buttons have different functionalities for each mode;
  + In the predefined shape mode, buttons 1, 2 and 3 are used to select the square, rectangle and triangle shape, respectively.
  + In on-the-fly mode, all of the number buttons except for 5 are used to orient the segment vertically, horizontally or at an angle. The button 5 is used to activate or deactivate MEMS mode, in which the STM32F407 is used to dynamically control the 3D printer based on the pitch and roll angles of the board.

In numbers mode, each number on the keypad represented the number to be drawn.

## LCD Display

The LCD display is used as a feedback device to the user to indicate what state the system is in and to keep track of what inputs were given to the system. Simple methods given in the LCD library of the STM32F429 were used to draw on the display. Using a coordinate system, lines and shapes can be drawn on the LCD by giving the proper coordinates needed for the chosen line or shape. Zero point on the y-axis on the LCD was defined to be the very top point of the display. The y-axis on the LCD was reversed in order to match the (0, 0) point on the whiteboard. For instance, to draw a line on the display, two locations on the coordinate system were given, and a straight line would be drawn between those points. Figure 5 shows several screenshots of the LCD display.

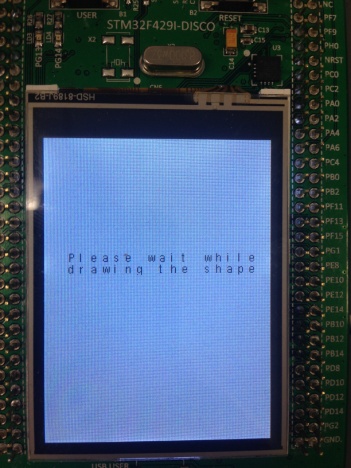
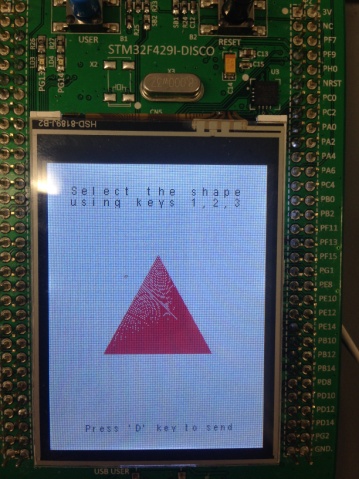
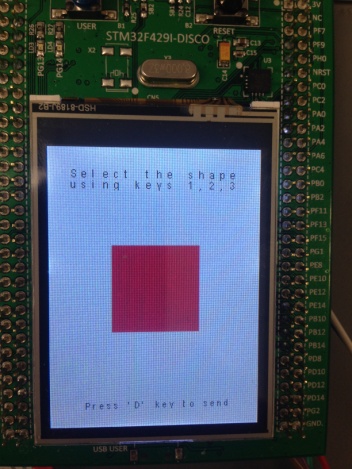
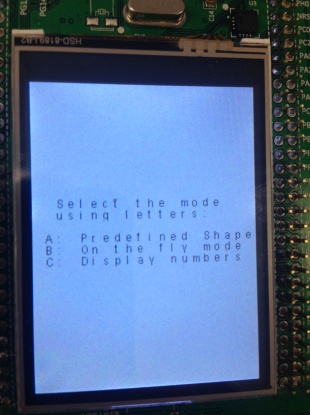


Figure 5: LCD Screenshots

To implement the blinking effect on drawing the next line, a simple technique of re-drawing the same line on top of the previous line with the background color was used. This technique gives the visual effect of the line appearing for a small period of time and then disappearing, although technically, a new line is being drawn on top of the previous one with a different color. Using this blinking effect, the user is able to see where the next line will be drawn. The results of how similar the LCD and whiteboard drawings are is shown in Figure 6 below.

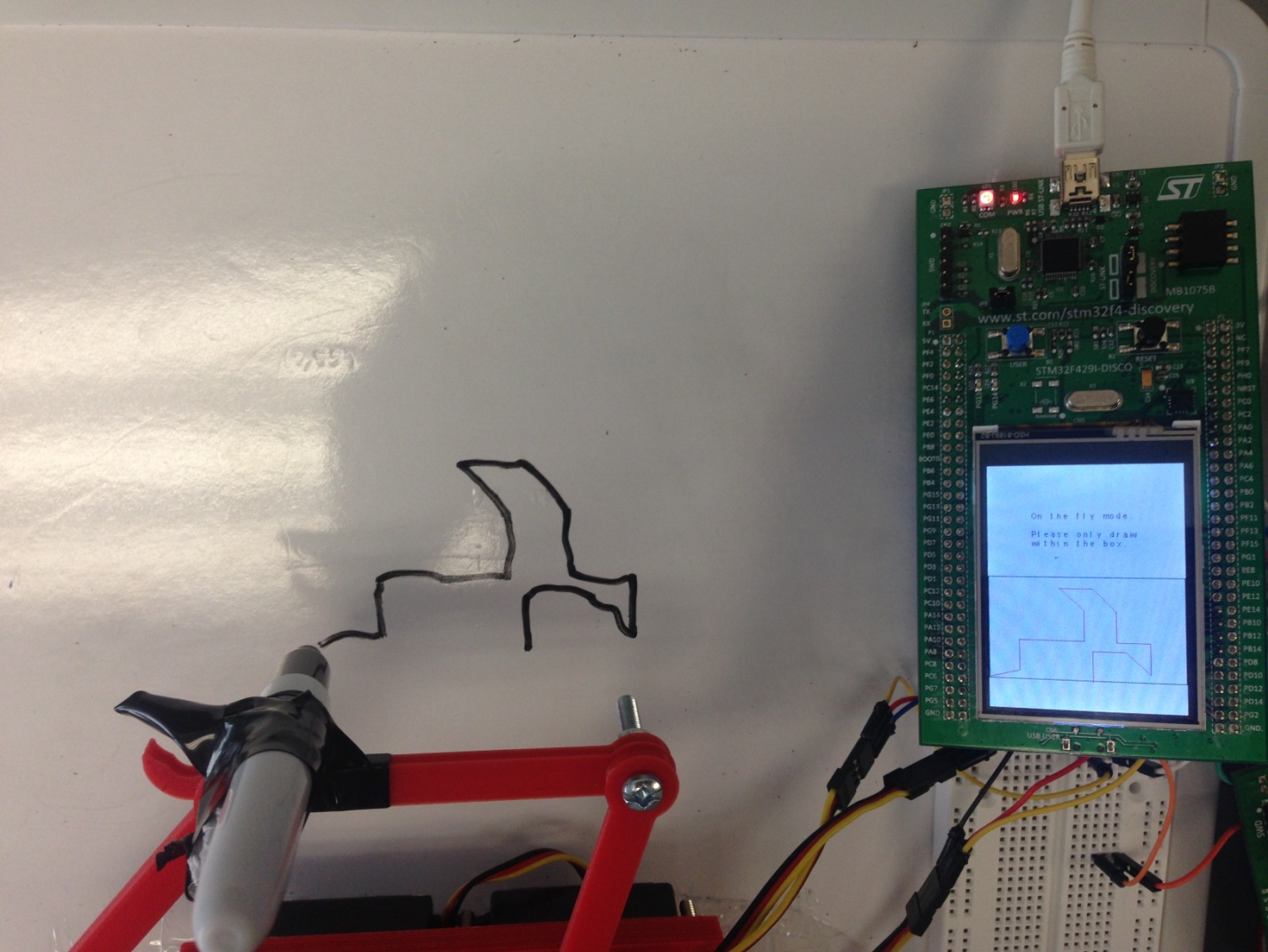


Figure 6: Whiteboard versus LCD drawing comparison

## Wireless Communication

In order to allow for communication between the transmitter (command) and receiver (3D printing) boards, the CC2500 RF transceiver chips (found on the TI MSP430 chipset) were used, in order to establish a means for wireless communication. The wired communication between the STM microcontrollers and the wireless chips was to be done using an SPI interface. To that end, the CC2500 driver which provides an abstraction for the lower-level SPI drivers was implemented. The CC2500 driver draws heavily from the MEMS driver which had been previously used in labs 3 and 4. At the lowest level, the wireless driver provides facilities for writing to and reading from the CC2500 registers. At a higher level, functions for changing the chip’s mode (e.g. receiver, transmitter, idle modes), checking the size of the receiver/transmitter buffers, flushing these buffers and reading/writing data from/to the receiver/transmitter FIFOs were implemented. In order to ensure the proper functioning of the wireless communication, it was essential to assign the configuration registers with the appropriate values. Most of these values were provided (with the others being set to their default values). The 2.4 GHz wireless channel was configured to 32, as per the requirements (i.e. group number (4) \* 8). The driver also includes a routine for low-level initialization, which is used to set the various GPIOs and SPI structure and enable hardware clocks. At the highest level, the modules which read and write data to/from the FIFOs, once such data is available (e.g. once a user selected a shape to draw, once a packet is received), were implemented.

## Transmitter OS Threads High Level Design

Three different threads are running at the same time on the transmitter board. These threads are the keypad thread, LCD control thread and finally the data send thread. The interaction between the threads, the interaction between the modules of the transmitting side as well as the interaction between the receiving and transmitting system are shown in the block diagram in Appendix 1. The priorities were set so that the data send thread has the highest priority, LCD control thread has normal priority and the keypad thread has the lowest priority. The reason for that is because both the LCD and data send threads can only be executed by the keypad thread, whenever there is a user input. When there is an input, the response from the system is more important than listening to the next instruction. So whenever there is a need to send data, this happens first and then any information is indicated on the LCD, and finally, the keypad can be listened to again. All of the threads were initialized with 600kB default stack size to guarantee safe thread operations as shown in Figure 7.

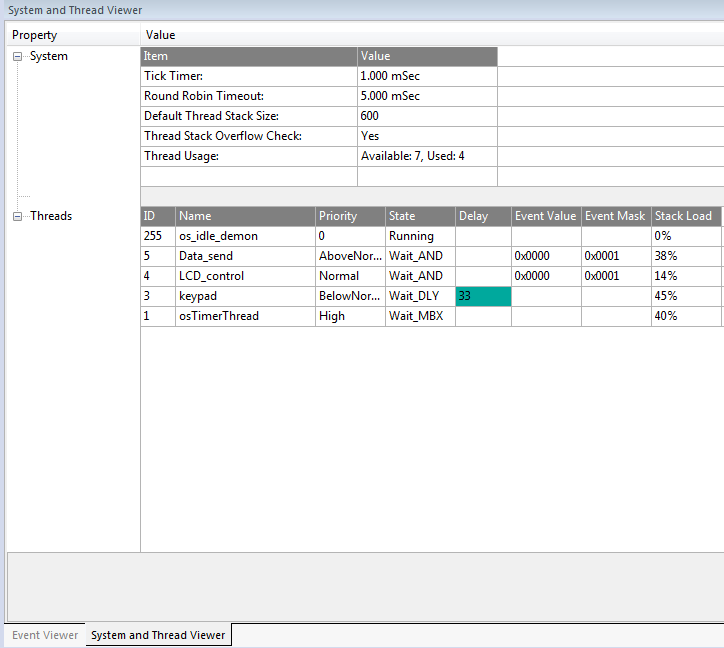


Figure 7: Thread priorities and stack sizes

### Keypad Thread

The keypad thread acts as the "master" thread that controls the other threads. In fact, the only independent thread in the system is the keypad thread, meaning that it is the only thread to be executed periodically by a timer instead of waiting for a signal. There only needs to be a response from the system if there is an instruction from the user, and this is why the keypad is used to send signals to the LCD and data send threads. The keypad thread periodically checks the buttons to see if there are any inputs, and then if there are, it modifies the global variables affected by the input, and then sends a signal to the proper thread.

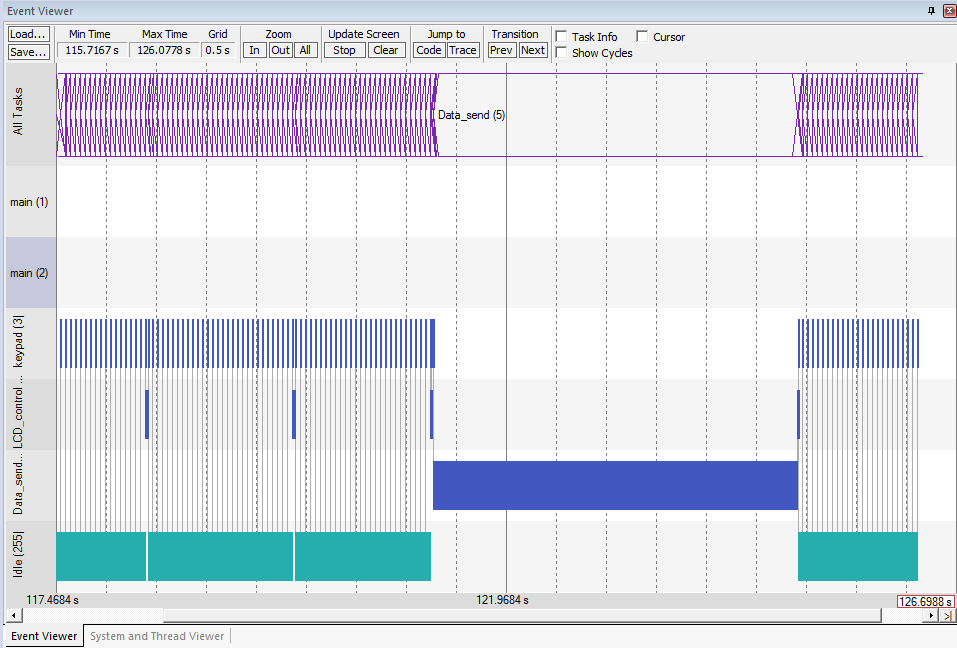


Figure 8: Mode A (predefined shapes) event viewer

As it can be seen above in Figure 8, the keypad thread executes periodically. Whenever a button is pressed on the keypad, the LCD thread executes for a little bit to print new information on the display. Finally when something needs to be drawn on the whiteboard, data send thread takes over and sends the data wirelessly to the receiver.

### LCD Control Thread

The LCD control thread controls what is being shown on the display of the transmitter board. It waits for a signal from the keypad thread, and when the signal comes, it checks the appropriate global variables to decide what should be shown on the display. First, it checks which mode the system is in and which mode the system was in the previous state. If the mode has changed, it clears the display and prints the current mode's home screen.

In mode A (i.e. the predefined shape mode), the only occasion when the display needs to be refreshed is when the user is scrolling through the shapes. In mode B (i.e. the on-the-fly mode), however, a blinking effect has to be implemented to indicate the position of the next line to be drawn if the enter button is pressed. To do so, a signal is sent to the LCD control thread periodically to show the effect.

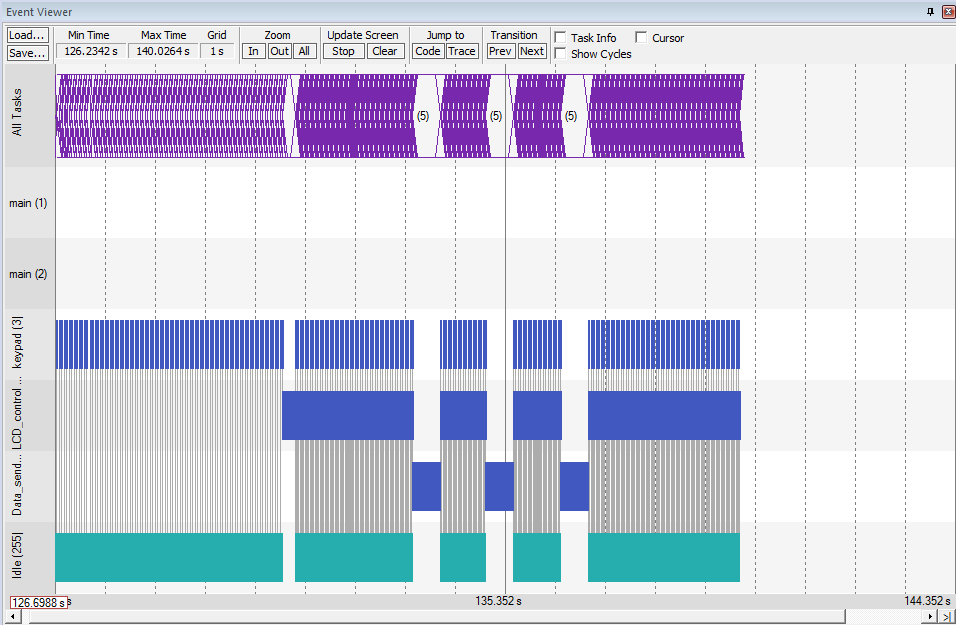


Figure 9: Mode B (on the fly) event viewer

As can be seen in Figure 9, when mode B is selected, the blinking effect the LCD control thread starts executing a lot more often compared to mode A shown in Figure 8.

### Data send Thread

The data send thread executes only when data needs to be sent. It waits until a signal is received from the keypad "enter" button, and when the signal is received, it checks the global array which holds the variables needed to be sent and sends it to the receiver board.

Although sending the data takes very little time, the data send thread holds control for a lot longer than that. This forbids the user to give more instructions while there is a drawing in progress. For example, as evidently shown in Figure 8, it takes a long time to fully draw one shape. After a "please wait" message is displayed on the LCD, the data send thread locks the system to prevent the user from giving any other instructions. Any inputs given during this time will be ignored by the system. Comparing Figure 8 and Figure 9, it can be seen that on-the-fly mode holds the lock for a smaller amount of time because of the fact that drawing a single line takes less time to complete than drawing a full shape.

## Receiver OS Threads High Level Design

On the receiver side, 3 threads were implemented, namely the data receiving thread, the servomotor thread and the MEMS thread. Communication (and synchronization) of the threads was done using message queues which share their source of memory from a single memory pool. The interaction between the threads, the interaction between the modules of the receiving side as well as the interaction between the receiving and transmitting system are shown in the block diagram in Appendix 1.

### Data receiving Thread

This thread is responsible for periodically checking whether a data packed was received. The system was designed such that a data packet always contains 4 bytes, with the first byte being a preamble (also used to signal that the MEMS mode was to be used), followed by 3 bytes, one per coordinate, the x-coordinate, the y-coordinate and the z-coordinate (in this order). Once it is determined that a packet was received (i.e. the RX FIFO contains at least 4 bytes), the bytes are read and the required action is taken. If the preamble byte is found to be equal to 254, the mems mode flag is set, allowing for the MEMS thread to know whether it should be functional. Otherwise, if the preamble is equal to 255, the data which was read from the FIFO is then sent to the servomotor thread via a message queue. To implement the periodic polling, a TIM3 clock, running at 20 Hz, was used. When the clock expires, it signals this thread, allowing it to become active and proceed with its actions.

### MEMS Thread

This thread is used in order to implement the MEMS mode, which can be chosen by the user using the procedure outlined above. The functioning of this thread is similar to the previous labs. An external interrupt on a line connected to the MEMS was setup. Once an interrupt occurs and if the MEMS mode flag is set, the normalized accelerations are read and used to calculate the board’s pitch and roll angles, as was done in previous labs. The calculated values were then filtered using 2 Kalman filters (one for each angle), using the procedure used and discussed in all 4 of the previous labs. The normalization of the accelerations was done using an (offline) calibration procedure which was outlined in [2]. Once the filtered values are retrieved, they are converted to x,y positions as follows:

|  |  |
| --- | --- |
|  |  |
|  |  |

The idea was to map the possible angles’ range of 0 ̊ to 180 ̊, to the range of possible x,y positions reachable by the drawing arms. Note also that in the case of the y-coordinate (i.e. roll angle), angles below 45 ̊ were set to 45 ̊ and angles above 135 ̊ were set to 135 ̊, in order to make the user experience more pleasant (i.e. the user doesn’t need to tilt the board too much, which can be difficult with all the wires, in order to reach the full range of y coordinates).

Finally, once the positions are determined, they are packed into a message queue and sent to the servomotor thread.

### Servomotor Thread

This thread is responsible for issuing the commands needed for moving the drawing arms to the correct position. First, this thread initialized the motors and timer/PWM structures required for the proper functioning of the motor. In the main body of the thread, it waits for a signal to be set. This signal is set by either the data receiving thread or the MEMS thread. Once the signal is set, it checks on which message queue data was received (i.e. this queue should match the thread which most recently set the signal). The message, containing the new position to which the arms must be moved, is then read and converted to angles (using the method described in the theory section), which are then used to set the PWM duty cycle for the motors.

# Testing and Observations

As the system is composed of numerous components, testing of each component was done separately. Testing for the motors, MEMS, keypad and LCD was done in a straight-forward way. Once each of these modules was completed, they were tested individually to ensure that their functionality matches our expectations. For example, the keypad was tested to ensure that the keys are properly mapped and that the debouncing is working properly. The LCD was tested to ensure that the messages and drawings shown on the screen are in line with the user’s input. Similarly, the motors were tested by programming them to move to various predefined positions and making sure that they reach these positions with an acceptable degree of accuracy.

The testing of the wireless communication component was done in a number of steps. First, once the basic SPI read/write functionality was implemented, various values were written into the configuration registers and then read them back, to ensure that these 2 core functions work properly. Once this was validated, tests were made to make sure that the receiver board was able to read the beacons transmitted by the board which was setup in the lab by the TAs. Finally, the transmitter module was written, the required wireless channel was modified to 32 (from the default 0) and it was verified that test messages, sent by the transmitter board, were properly received and read by the receiving board.

Once all the individual components were shown to work properly, all of the components were integrated into a single system. Extensive testing on the entire system was performed, making sure that all the modes and their functionalities work according to the specifications.

## Breakdown of Work between Team Members

|  |  |
| --- | --- |
| Name | Work Description |
| Olivier Laforest | * Wireless * Motor calculations |
| Genevieve Nantel | * Hardware * Motor configuration and calculations |
| Nuri Ege Kozan | * LCD, keypad and transmitter board logic * Combining all components |
| Maxim Goukhshtein | * Wireless, MEMS and receiver board logic * Combining all components |

# Conclusion

In conclusion, this report has shown how a 3D printing device has been implemented using two separate subsystems, a transmitting and a receiving subsystem, which communicate wirelessly. The transmitter system was realized using the STM32F429 Discovery board, an alpha numeric keypad and a TI CC2500 RF wireless chip, which recorded the user's input, provided visual feedback through the LCD display of the STM32F429 board, and processed and transmitted those commands wirelessly to the receiver system. The receiver system was built from a STM32F407 Discovery board, a TI CC2500 RF wireless chip, three HiTech HS422 stepper motors and a set six 3D printed plastic parts. The receiver system successfully converted the user commands received wirelessly to appropriate printer motions in order to print the user's selection. This project is an example showcasing how MCU evaluation boards combined with a small number of components can be combined to create complex systems.

# References

|  |  |
| --- | --- |
| [1] | Tilzor, "Library 33-PWM for STM32F4," 7 September 2014. [Online]. Available: http://stm32f4-discovery.net/2014/09/library-33-pwm-stm32f4xx/. [Accessed 14 April 2015]. |
| [2] | STMicroelectronics, "AN3182 Application note: Tilt measurement using a low-g 3-axis accelerometer," *2010.* |

# Appendix 1

Block Diagram of 3D printing system

kalman\_filter.c

servomotor.c

mems.c

Wireless link

threads.c

servomotor\_thread

mems\_thread

rx\_thread

main.c

main.c

wireless\_rx.c

cc2500.c

threads.c

cc2500.c

wireless\_tx.c

Data\_send\_thread

LCD\_control\_thread

keypad\_thread

keypad.c

LCD\_draw.c