Team Twenty Four
Project *24 Hands*

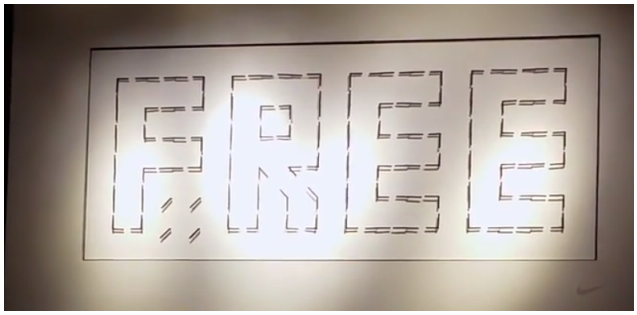
By *Damini Chopra (dachopra@bu.edu), Konstantino Sparakis (sparakis@bu.edu), and Siddhesh Kulkarni (siddhesh@bu.edu)*
BU ENG EC535 - *Introduction To Embedded Systems*
Dr. Ayse K. Coskun
12/11/2014

I. Abstract

In this project we attempted to create a seven segment display, using stepper motors and moving hands. We wanted to create this because we saw it as a fun challenge, that would enable all of us to learn a lot. We were able to complete perfectly timed software which communicating through gpio pins and our logic circuit design was able to stop the motors with the hands pointing in the proper direction to display a number or letter. We were also successfully able to interface the TI - Chronos EZ430 watch with our project to send gesture commands along with sensor data to a gumstix in order to display different things on our seven segment display board. Unfortunately, we miscalculated the need of a stronger power supply, meaning not all twenty four stepper motors could be on at the same time, and we ran into some wiring issues as well. But regardless of this we were able to get the stepper motors to work at a smaller scale and once the remaining issues are resolved then our solution will work as intended.

II. Introduction

We originally were very confused as to what project we were going to do. We had planned to do triangulation using Bluetooth on a car, but it seemed evident that every group was using a car for something and not all of our group members were too excited about the project. Konstantino walked into the Nike store on Newbury in Boston one day, and they had this display made up of several clocks displaying letters and the time, he video tapped this and shared it with the team. Everyone thought it was cool, and being engineers we started to discuss how this display was made. We discovered the name of the original piece, [1] *A Million Times – By Human since 1982,* Everyone showed more enthusiasm discussing about this display using clocks, and since we all wanted to work hard on something we all really liked, we decided to change our project to creating a replica as all team members were more engaged and it seemed fun and a very complicated task to accomplish.



In the big picture this project combined many different aspects of engineering that didn't seem very evident at first, from electrical engineering for the wiring, power supply, and logic circuit. Also mechanical engineering skills for designing and 3D printing the clock hands, and all the work to use the proper tools to create the wood backboard with proper incisions. It even included some artistic design skills when it came down to making it look aesthetic, choosing colors and painting everything. But mainly the embedded system skills came to be very important, using the gumstix along with a raspberry pi to use gpio pins and software to make everything work as it is intended to.

We needed a method to communicate with the board as to add some more embedded systems knowledge into our project and this is where the TI watch came into play. The TI watch contained an accelerometer and had other sensors , so it was decided to use this as an external sensor that the board would respond and react to accordingly.

In the end we were able to learn a lot about all the components that came together to create this project, but we also had some hiccups and problems on our way causing us to simplify the project at some points, such as using a single hand instead of two clock hands like the original in order to avoid the complexities of gear ratios. We also miscalculated the need for a stronger power supply meaning the motors did not receive enough power to move all together synchronously.

III. Design Flow

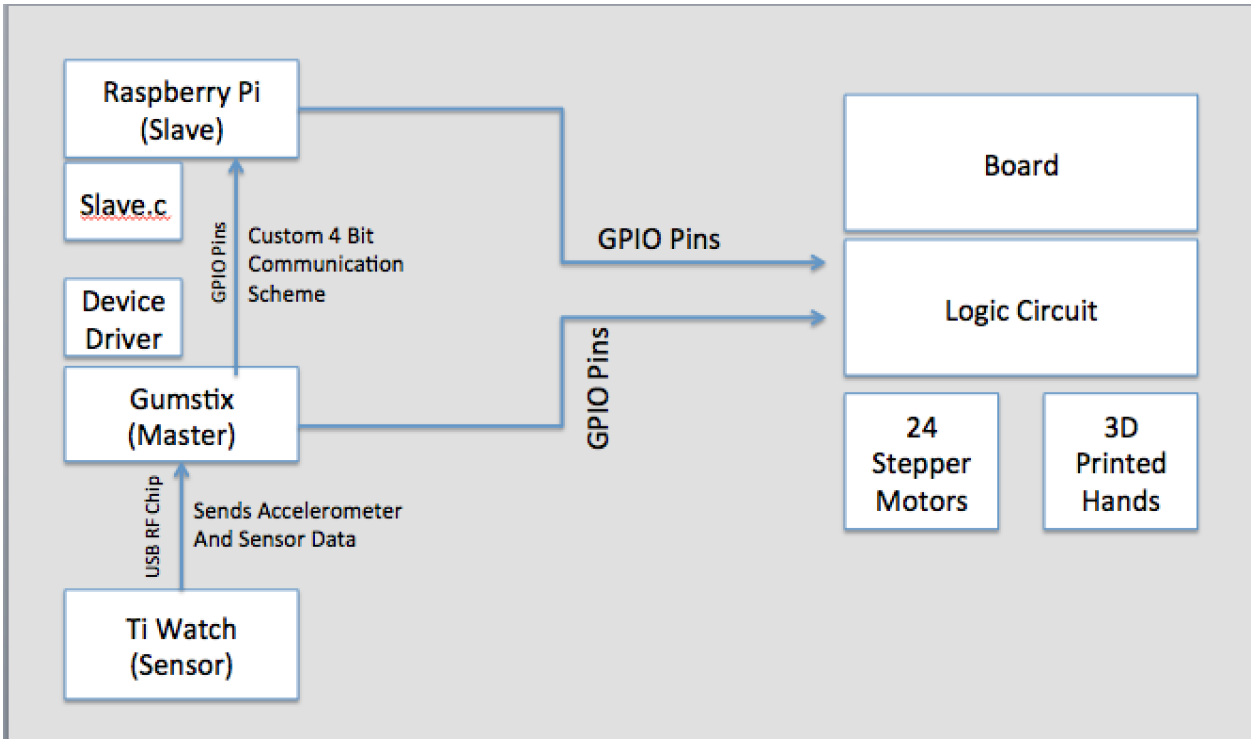


Figure 1: Flow chart of design

Components:

a. Gumstix
   i. *Description: The gumstix is the master and the brain of the operation, It has a device driver that controls everything and communicates with everything.*
   ii. *Worked on by: Siddhesh and Damini*

b. Raspberry Pi
   i. *Description: The raspberry pi receives commands over a custom 4 bit communication scheme to turn on and off certain gpio pins. It is a slave as no real computation happens there. Slave.c is the code that receives and handles signals.*
   ii. *Konstantino handled all raspberry pi related tasks, such as installing OS and writing code.*

c. Custom 4 Bit Communication Scheme

i. *Description: predefined signals sent every millisecond.*
ii. *Konstantino created slave.c and made the protocol working with Damini who implemented it on the driver side.*

d. Ti Watch
   i. *Description: Sends accelerometer readings and PPT readings to the gumstix over a RF USB chip included with the watch.*
   ii. *Siddhesh and Damini along with the help of Ozan were able to get this working and created a basic driver to get the gestures recognized. They then implemented the code into the main driver.*

e. Logic Circuit
   i. *Description: Designed to run all 24 motors synchronously and stopping a motor with a kill signal*
   ii. *Designed By Damini and implemented as a circuit by Siddhesh and Damini both obtaining the gates and soldering it together.*

f. Board
   i. *Description: Plywood, drilled and spray-painted. Holds the logic board, 24 stepper motors, 3D printed hands, Gumstix and Raspberry Pi.*
   ii. *Woodwork and spray-painting was done by Konstantino at the epic lab. As for the soldering and wiring was a team effort following the lead of Siddhesh and Damini who had a better understanding of the circuit.*

g. 24 Stepper Motors
   i. *Description: The model used was 28BYJ - 48 [2] and we used the* ULN2003 driver board.
   ii. *Konstantino found the model and drilled them into the board, while Siddhesh and Damini got them to work with a basic code.*

h. 3D Printed Hands
   i. *Description: These hands were designed by cad. Printed and spray painted back to fit on the stepper motors.*
   ii. *We came up with the cad design and sizing as a team, then Konstantino had them 3D printed and he removed the support material and spray painted them.*

IV. Project Details

a. Gumstix & Device Driver (See dig.c and ul.c for source code)

As working on gumstix as the master controller was the integral part of the project, we started working on the same. Use of 24 motors meant controlling it with 96 GPIO lines. So we decided to design a Logic Circuit. Even after making the logic circuit, we still needed 24 GPIO pins to control 24 stepper motors. The four signal pins used to drive one motor was brought down to 2. So total the project required 26 GPIO pins. In order to do that, we started searching for the maximum number of GPIO pins a gumstix can have. We searched the datasheets and found out the various GPIO pins. We then individually tested each and every gumstix GPIO pins. We realized that there are around 19 GPIO pins, which can be used for our project. As a result we started developing logic around it. We developed a code which operated different stepper

motors and also decided how many steps each stepper motor will have (which was a non linear equation). We also found out what will be the number of rotations required by a stepper motor from its current position in order to reach the final position. We made a switch loop from each and every angle and number of steps required. Following is snippet of the same,

```
case 0: stepper=0;
                    break;
 case 45: stepper=325;
                     break;
 case -315: stepper=325;
                     break;
 case 90: stepper=650;
                     break;
 case -270: stepper=650;
                          break;
```

As the number of GPIO pins were still less and we have just one gumstix, we decided to shift to Raspberry Pi as a slave microcontroller.

b. Raspberry Pi (See Slave.c, Setup.php, Makefile, and run.sh for the source code)

We decided to use a Raspberry Pi when it became evident that there were not enough gpio pins on one gumstix and with a gumstix shortage we opted out for the Raspberry Pi. The Raspberry Pi is also a lot easier to interact with, as you can plug it into a power supply and ethernet and simply ssh into it and do work.

After studying the GPIO pin layout and alternative functions of each pin, It was discovered pins 26, 27, 28 , and 29 where the most suitable for receiving the Custom 4 Bit Communication signals as pins 27 and 28 could not be used as outputs. The rest of the pins were set as output pins, the idea is that each output pin is on and connected to a specific stepper motor, when the pin is turned off , the logic circuit stops the motor.

On each start up the pins were reset to their original functions, so Konstantino created a quick PHP script, Setup.php that sets all the pins to their proper input output stage required by the project. The PHP script simply executes commands on the shell in a for loop that set all the gpios to their proper modes that we require.

We used Wiring Pi [3], which is a simple library for controlling gpio pins on the raspberry pi to implement the c code without too much of a hassle. Using this made compiling a little tricky so we created a makefile and a shell script, the shell script simply runs ./Setup.php , Make and ./Slave.c in that order in order to make it easy for others to use.

Slave.c is the code that executes in a loop reading whatever data is coming in through our Custom 4 Bit communication scheme and changes output pins accordingly.

c. Custom 4 Bit Communication Scheme (see Slave.c and for source code)

In ***Table 1*** we have all the 4 bit signals sent and what is meant to happen for that specific signal. We communicate by sending a signal every Millisecond. In order for this to work we use 4 output pins from the gumstix and 4 input pins on the Raspberry pi.

Table 1

4 Bit Signal	How the signal is handled
0000	Do Nothing
0001	Turn GPIO pin 11 Off
0010	Turn GPIO pin 13 Off
0011	Turn GPIO pin 15 Off
0100	Turn GPIO pin 19 Off
0101	Turn GPIO pin 21 Off
0110	Turn GPIO pin 23 Off
0111	Turn GPIO pin 29 Off
1000	Turn GPIO pin 31 Off
1001	Turn GPIO pin 33 Off
1010	Turn GPIO pin 35 Off
1011	Turn GPIO pin 37 Off
1100	Turn GPIO pin 40 Off
1101	Do Nothing
1110	Turn All GPIO Pins Off
1111	Turn All GPIO Pins On.

d. TI Watch EZ430-CHRONOS

The TI EZ430 - CHRONOS watch consisted on an in-built accelerometer which continuously transmitted its readings in ACC mode. The watch wirelessly communicated with the gumstix with its RF frequency band (915 Mhz) known as Simpliciti. Initially in order to communicate with the gumstix, the program first needs to transmit a sequence of setup commands. After transmitting the setup btes knows as START_AP bytes in the

code, the watch sends an acceptance signal, which sets up the communication between gumstix and the watch. After it is important to set up the input data rate, output data rate, baud rate, etc for the watch. Once again in order to start the communication the in “ACC” mode, we need to setup a sequence of bytes once setup, we start getting the readings of the accelerometer. Depending upon the values of accelerometer in x, y and Z direction, we can assign different tasks to different values. If we start the watch in “PPT” mode, then depending on the duration of pressing the button, we can assign a different command from the watch (which we assigned to display time).

e. Logic Circuit

The major problem was to reduce the number of GPIO pins. This led to devising a multiplexing circuit for the same. We found out various ways and then finalized to the logic circuit explained below:

The major problem was that each stepper motor would take different number of steps. As a result, there should be some control pins which will drive each stepper motor. We also found that the different inputs which we gave to the four pins of each stepper motor had a logic. The third and the fourth signals were compliment (NOT) of the first and the second respectively. As a result we decided to use a NAND gate IC as a NOT gate to generate the compliment of first and second input pins. We also realised that in order to control each stepper motor, we can use an AND gate IC which will control the output to the motors depending on its control signal pins. The IC will be ON only if we send a control pin to the motor. As a result we finalized that we will require 12 AND gate IC along with 1 NAND gate IC.

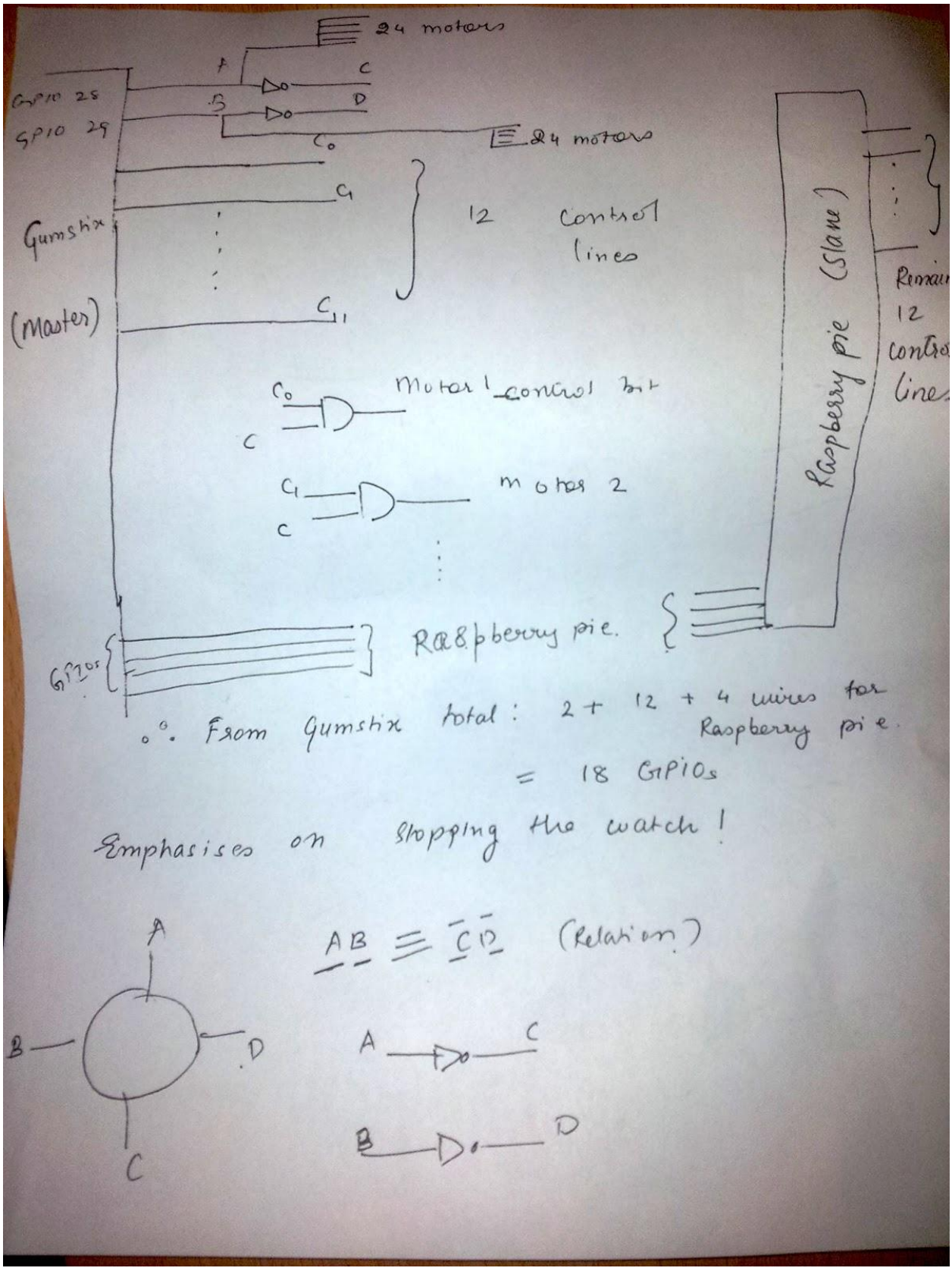


Figure 2: This is the logic circuit implemented

f. Board

The board was decided to be 2ft x 4ft because we wanted our design to be big enough to be visible from some distance and to make an impression. Over thanksgiving break with access to a car Konstantino was able to go to Home Depot and buy the 2ft x 4ft Plywood, Two 2ft x 5 inch side woods, screws for everything, white spray paint and angles to hold the side woods up (***Figure 3***).

First the wood boards where all spray painted. Afterwards, we measured the motors size and 3/8th's of an inch holes were drilled into the wood at the Epic lab using a drill bit and the machinery there, following the perfect dimensions to account for 24 motors and the 5 inch span of each hand (***Figure 4***). after this all 24 motor were drilled in along with the side wood boards which are there to create distance between the board and a wall for there to be room for all the electronics in between.

Figure 3: buying supplies

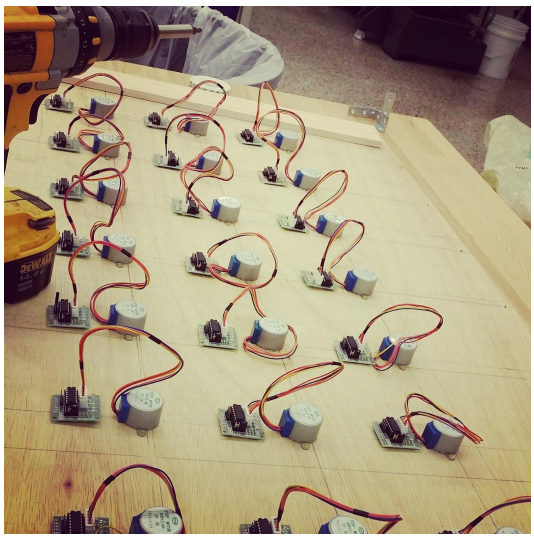


Figure 4: All stepper motors drilled in.

g. 24 Stepper Motors and Drivers

When we were looking for something to move the hands we had the idea of obtaining already fully working clock hands and hack them to work for our project. We were quickly concerned that we might not be able to successfully hack a clock fixture to do what we needed it to, and so we opted out to getting stepper motors. We looked for a small cheap motor that had enough power and speed to move the hands, and so we found the 28BYJ - 48 [2] bundled with the ULN2003 driver board [4]. What the driver board does is create a safety barrier between the microcontroller and the stepper motor as not burn either out.

We suspect that some of the driver boards may not be working as intended and also the stepper motor is slower than we expected it to be and also tends to heat up excessively but at the same time was just good enough to accomplish our give task.

h. 3D Printed Hands

For the design of the 3D hands we had originally planned to have two hands per stepper motor. It quickly became evident that our lack of knowledge in mechanical engineering and gear ratios made this very difficult. We opted out to change our design to a one hand per motor, where the hand could reach all the way to the next motor, which is 12.7 cm in length. This gave us the issue where hands could collide so we decided to have four different heights that arranged properly would avoid this collision (***Figure 5***). Once we had the size and dimensions (***Figure 5***) done Konstantino went to the Epic lab where they helped him create a CAD design (***Figure 6***) and get it 3D printed. Once 3D printed one has to remove the support material which is used to support the printing. After this Konstantino spray-painted them black as they were originally printed white. This was done to add a contrast between the board and the hands for aesthetic purposes.

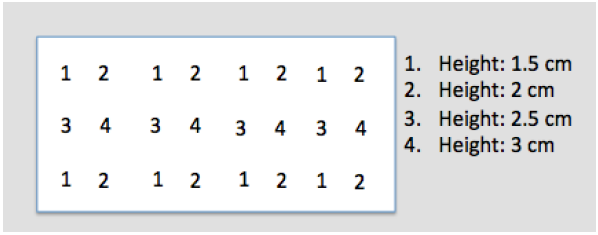


Figure 5: Height layout of each hand and key

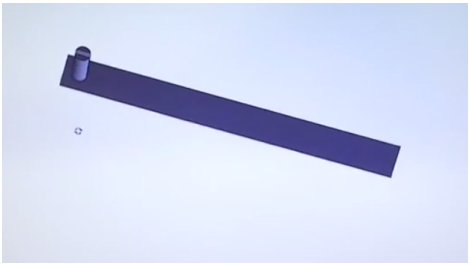


Figure 6: CAD Design of hand

V. Summary

In conclusion, creating a seven-segment display using stepper motors was a lot more difficult than any one of us expected it to be. We had to create a digital logic circuit that would allow us to synchronously have all stepper motors moving at the same time, and with a kill signal being able to turn off a specific motor. In order to implement the kill signal we needed more gpio pins than the gumstix could supply so we created a four bit communication scheme between the gumstix and a raspberry pi, telling the raspberry pi which gpio pin to turn on or off. Also this project required craftsmanship in wood, along with learning how to design things with cad and 3D printing them. We spent a lot more time than expected soldering and debugging. We were able to accomplish all this along with having software that did exactly what we needed, commanding which gpio pin to turn on exactly when we needed it to get the display to work properly when interfaced with commands from the TI watch.

We all learned a lot from implementing all these different components and unfortunately our end product was not able to move all twenty-four stepper motors synchronously because we miscalculated the need for a stronger power supply. This is our remaining challenge and we hope to debug this issue and make all our effort pay off with a working final project.

IV. References

[1] A Million Times--By Humans Since 1982.
http://www.humanssince1982.com/a-million-times/

[2] Data Sheet for 28BYJ - 48 Stepper Motor. http://robocraft.ru/files/datasheet/28BYJ-48.pdf

[3] Wiring Pi. http://wiringpi.com/

[4] Data Sheet for ULN2003 Driver. http://www.farnell.com/datasheets/1690348.pdf

[5] Wiki for TI watch used for help.
http://processors.wiki.ti.com/index.php/EZ430-Chronos?DCMP=Chronos&HQS=Other+OT+chronoswiki

[6] Data sheets and Info on TI watch.
http://www.ti.com/tool/ez430-chronos&DCMP=Chronos&HQS=Other+OT+chronos

[7] Guide used for controlling stepper motors.
http://www.elprocus.com/stepper-motor-control-using-avr-microcontroller/