CodeValue ClockWall

Design Spec

**Version 1.0 (preliminary)**

December 2015

By Allan Schwartz, Eli Arbel



### Document Version Tracking

|  |  |  |  |
| --- | --- | --- | --- |
| Author | Date | Version | Comments |
| Allan Schwartz | December 2015 | 1.0 | As-implemented |

Contents

[Document Version Tracking 2](#_Toc439238213)

[Table of Figures 2](#_Toc439238214)

[Definitions, Acronyms and Abbreviations 4](#_Toc439238215)

[Introduction 5](#_Toc439238216)

[About The Clock Wall 5](#_Toc439238217)

[Single Frame 5](#_Toc439238218)

[Single Train 5](#_Toc439238219)

[Sequence Diagrams 6](#_Toc439238220)

[Therapy Train Diagram 6](#_Toc439238221)

[Control Message Sequence Diagram 6](#_Toc439238222)

[Packet Formats 6](#_Toc439238223)

[ECU Code Review 6](#_Toc439238224)

[IVI Code Review 7](#_Toc439238225)

[Summary 10](#_Toc439238226)

[Why We Remove Dead Code 10](#_Toc439238227)

[Robustness 10](#_Toc439238228)

[Efficiency 10](#_Toc439238229)

### Table of Figures

[Figure 6: Single Train 4](#_Toc439238230)

[Figure 7: Therapy Train Diagram 5](#_Toc439238231)

[Figure 8: Control Message Sequence diagram 5](#_Toc439238232)

### Definitions, Acronyms and Abbreviations

|  |  |  |
| --- | --- | --- |
| Term | Description | Comments |
| Arduino |  |  |
| Frame or Packet | A series of bytes which comprise one message |  |
| H-Bridge | A motor-driving circuit |  |
| Step-Motor | A motor capable of moving only 1° (1/360 of a revolution), or less, per step. |  |
| I2C | I2C (Inter-Integrated Circuit) is a serial communications bus standard used typically to attach peripheral chips to microcontrollers. |  |
| TWI | Two Wire Interface, another name for I2C |  |
| SPI | SPI (Serial Peripheral Interface) is a synchronous serial communications bus standard, which is up to 100 times faster than I2C. |  |

*<reviewers: please add acronyms or terms or jargon used elsewhere in this document, which should be defined above>*

## Introduction

The purpose of this document

### About The Clock Wall

The ClockWall …

### Motor Controller

Blah, blah

### Row Controller

Blah blah

### ClockWall Controller

Blah, blah

## Detailed Design

### Clock Wall Substrate

Some thoughts on how the substrate is constructed, how the Motor Elements, and Motor Controller attach to it; how signal and power are routed, where the power supplies reside; how it is attached to the wall; the overall dimensions of this.

### Motor element

The Motor Element are the pieces that comprises one of the 288 devices. It includes

* PCB
* VID28 Stepper Motor, attached to the PCB
* Microcontroller, or MCU board, socketed
* Round encasing piece
* Front bezel
* Two clock hands

In looking at the various videos on YouTube, it appears that the Motor Element enclosure is made from White 3” or 3-1/2” or 4” plastic pipe. This is an extremely common building supply in America, used for all plumbing construction. It costs under $1 per foot, and is easily cut and bonded. This table summarizes the exact dimensions:

|  |  |  |
| --- | --- | --- |
| **Nom. Pipe Size (in)** | **Outside Dia.** | **Inside Dia.** |
| 3” | 3.50” (8.89 cm) | 3.042” (7.73 cm) |
| 3-1/2” | 4.00” (10.16 cm) | 3.521” (8.94 cm) |
| 4” | 4.50” (11.43 cm) | 3.998” (10.15 cm) |

I don’t know if these same dimensions can be found in Israel.

My vision of the Motor Element, is the pipe enclosure completely surrounds and contains the PCB, with stepper motor attached, a front bezel, and clock hands. A Motor Element would look sort of like this, in cutaway:

<figure to be drawn>

Still to be designed, is how the PCB attaches to the Enclosure, how the front bezel attaches, if there is a bottom or not, and if so, placement of holes for signal/power cables.

### Motor Controller Alternatives

The motor controller is probably the heart of the system, and careful design and implementation is important to the success of the project. The purpose of the motor controller is to control the clock hands, at very precise timings. There are have several designs suggested. Let me try to enumerate them:

1. Arduino microcontroller, 3 H-Bridges, controlling 1, 2 or 3 pairs of clock hands
2. Arduino microcontroller, no H-Bridges, controlling 1, 2 or 3 pairs of clock hands
3. No microcontroller (but communication logic), single H-Bridge, controlling 1 pair of clock hands

That is actually 7 separate designs. I believe a single MCU, with a TB6612 H-Bridge controlling 1 pair of hands is the preferred design, for 3 reasons:

1. The cost of the PCB is directly proportional to the number of square centimeters of size. A PCB controller 2 or 3 clocks (pairs of hands) would be quite expensive, just because it would be so large.
2. The complexity of controlling each pair of clock hands is fairly high. It may require four channels of PWM (pulse-width-modulated) per stepper motor, which requires several timer devices. The smaller Arduino do not have enough timers to a.
3. Typical board which controls stepper motors uses an H-Bridge, for a number of reasons.

### Large Network of Arduinos

Controlling a large number of Arduinos is an interesting problem. I propose that a “Row Controller” control the set of the Motor Controllers in one row (presumably 24). This Row Controller would issue commands to the Motor Controllers, under its command, in its row. The obvious way to connect these devices is over an I2C bus, also called a TWI (Two Wire Interface). 24 devices is well within the capabilities of I2C, including addressing, latency issues, and signal quality. The Row Controller would be the I2C Master, and the Motor Controller would all be I2C Slave devices, each with a different I2C address.

Then, the 12 row controllers are addressed directly by the ClockWall Controller. The ClockWall Controller might communicate with the 12 Row Controllers over I2C. However, there are certain technical problems with this, and better design might be to use another bus communication technology for ClockWall Controller to Row Controller. It is suggested the SPI bus might work. The SPI bus operates at a much higher speed, however is designed for short distance communication between integrated circuits on one circuit board. There are examples in the literature about how to overcome the distance limitation.

Because we are attempting to control a time-sequence animation, the timing of the step motor actions is important. It is envisioned that all the Motor Controllers can be synchronized to approximately millisecond accuracy; using just I2C commands that transmit the time register from the Row Controller to the Motor Controllers. Then the **millis()** function call on all the Row Controller’s would be the same, within the same millisecond.

Even more accurate synchronization could be done, with one extra signal from the Row Controller, pulsed from the Row Controller and received on each Motor Controller in a Digital IO bit tied to an interrupt. Then the Motor Controllers could be synchronized to within a microsecond.

Likewise, the ClockWall Controller can synchronize the Row Controllers in the same ways.

The commands to step a motor could either be immediate commands, or timed to done at a specific millisecond.

### Microcontroller Selection

**Motor Controller.**

Obviously, the Motor Controller is most numerous element of the design, requiring 288 of these MCUs. The selection is critical. We choose an 8-bit MCU because of cost; and we choose a member of the Arduino family because of ease of developing, ease of programming the devices, and the libraries. In selecting a small-form factor Arduino, or Arduino compatible, we have these choices:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Processor | SRAM/FLASH | Analog Input  Digital IO/PWM | Comm |
| Arduino Micro  48mm x 18mm | 8-bit ATmega32U4  16MHz | 2.5 k / 32 k | Ana: 12  Dig: 20/7 | I2C  SPI  Micro USB |
| Arduino Mini  *(retired)* | 8-bit ATmega328P  16MHz |  |  |  |
| Arduino Nano v2.x  45mm x 18 mm | 8-bit ATmega168  16MHz | 1 k / 16 k | Ana: 8  Dig: 14/6 | I2C  SPI (no lib.)  mini-B USB |
| Arduino Nano v3.x  44 mm x 18 mm | 8-bit ATmega328P  16MHz | 2 k / 32 k | Ana: 8  Dig: 14/6 | I2C  SPI (no lib.)  mini-B USB |
| Arduino Pro Mini  33 mm x 18 mm | 8-bit ATmega328  16 MHz | 2 k / 32 k | Ana: 6  Dig: 14/6 | I2C  SPI (no lib.)  6-pin header |
| MKR1000  *(coming soon)* | 32-bit Atmel SoC  48 MHz | 32 k / 256 k | Ana: 7  Dig: 8/4 | I2C  SPI  USB |
| Particle Photon | 32-bit STM32F205  120 MHz | 1024 k / 128 k |  |  |
| Sparkfun  Teensy 3.2 | 32-bit ARM  72 MHz | 64 k / 256 k | Ana: 21  Dig: 34 /12 | SPI, I2C, I2S, CAN,  IR, UARTs, USB |
| STM32F0 Discovery  *$9.44* | 32-bit ARM | 8 k / 64 k | Plenty | Plenty |
| MSP432 LaunchPad  *$12.99* | MSP432P401R  48 MHz | 64 k / 256 k  + FRAM | plenty | 4 I2C, 8 SPI, 4 UART |

I believe the first choice for the MCU for the Motor Controller is the Arduino Nano v 3.x. It can be found for between $5 - $10 from a number of different vendors.

**Row Controller**

The Row Controller has slightly more communication and memory requirements than the Motor Controller. It does not need to be piggybacked onto a Motor Element, and should stand alone – physically attached to the Clock Wall Substrate., but not to any Motor Element.

We have thought about using another Arduino Nano for this application. However, I fear that memory requirements are such that the Arduino is too limiting. Plus, the Arduino could not be both I recommend the MSP432 LaunchPad, listed above. At a price of $13 direct from TI (possibly less from a distributor), it is definitely a low-cost solution. Using TI-RTOS, it is very easy to build a threaded application. It has great support for Interrupt based I2C and SPI communication.

**ClockWall Controller**

The ClockWall Controller must communication with the Row Controllers. It must also serve as an IoT node, and provide a network service of some kind (to be designed), to the users or client computers. We should defer MCU selection, until the communications protocol is fully specified.

### Motor Controller Design

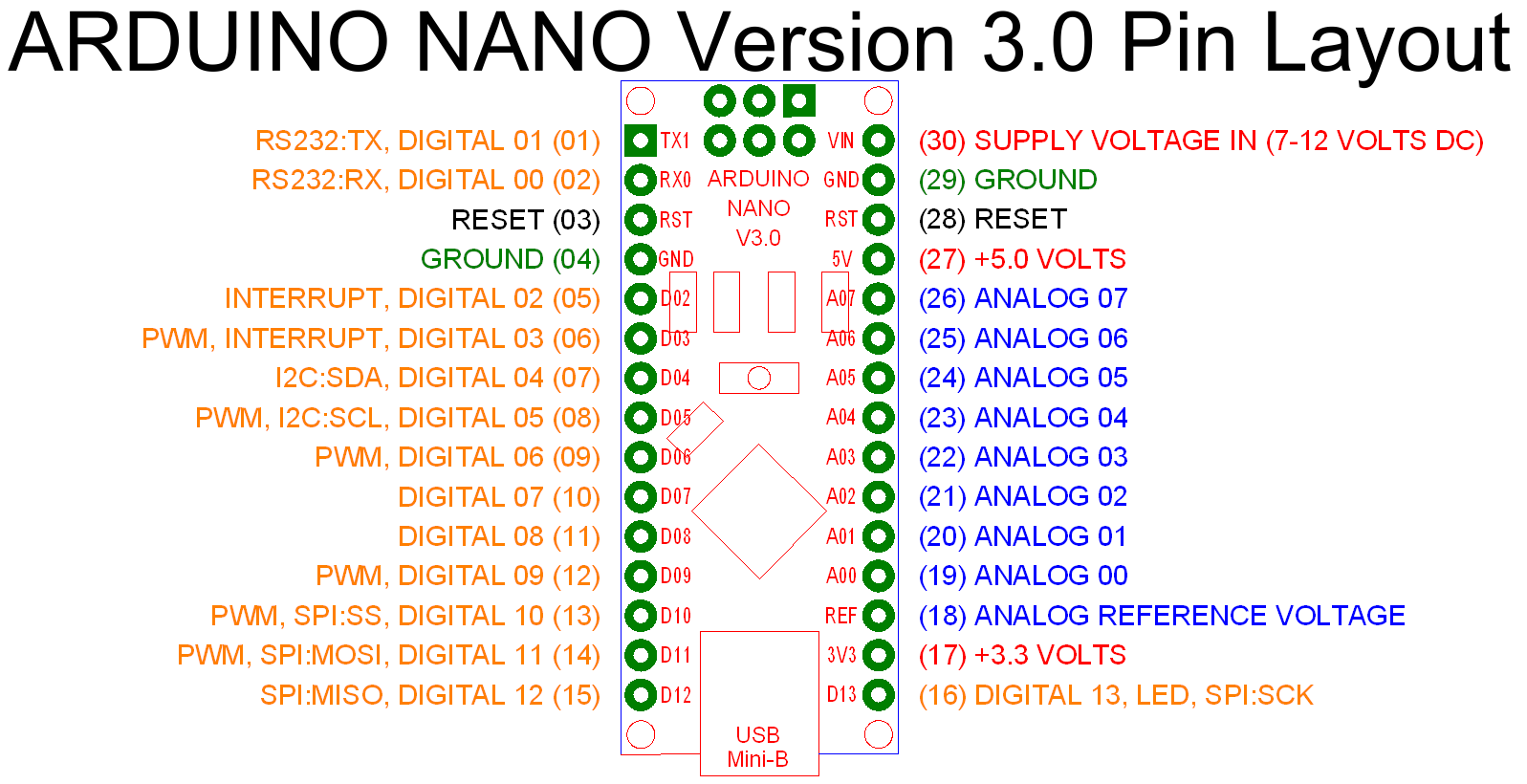
The Arduino would sit on a PCB substrate, plugged in. This reduces the risk of the overall project, because the PCB substrate is not nearly as expensive with just a processor connector as the processor itself.

Also on the PCB is

* A TB6612 dual H-Bridge chip
* An 8-bit DIP switch
* A header for 5 signals in: SCA, SCL, TIME\_SYNC, VDD, SIG\_GND
* A header for 5 signal out: SCA, SCL, TIME\_SYNC, VDD, SIG\_GND
* A header for 2 power signals: VM, GND

The PCB would need to slightly larger than the motor, which physically mounts to it. The Motor is 64mm x 35mm in size, so we can estimate that the PCB will be between approximately 80mm square.

**IO Pins on the Arduino Nano**



**IO pin allocation**

|  |  |  |  |
| --- | --- | --- | --- |
| Arduino Pin | Input/Output | Function | Other device |
| Digital 00 |  | Reserved for code download |  |
| Digital 01 |  | Reserved for code download |  |
| Digital 02 | I | Time sync interrupt | Time Sync |
| Digital 03 (PWM | O | PWM motor control | Motor Contact A1 |
| Digital 04 / I2C: SDA | I | I2C | SDA |
| Digital 05 / I2C: SCL | I | I2C | SCL |
| Digital 06 (PWM) | O | PWM motor control | Motor Contact A2,A3 |
| Digital 07 | O | Motor A / Motor B select |  |
| Digital 09 (PWM) | O | PWM motor control | Motor Contact A4/B4 |
| Digital 10 (PWM) | O | PWM motor control | Motor Contact B1 |
| Digital 11 (PWM) | O | PWM motor control | Motor Contact B2, B3 |
| Digital 12 (SCK) |  | Reserved for SPI |  |
| Digital 13 | O | heartbeat | LED |
| Analog 00 | I | Option bits | DIP switch bit 1 |
| Analog 01 | I | Option bits | DIP switch bit 2 |
| Analog 02 | I | Option bits | DIP switch bit 3 |
| Analog 03 | I | Option bits | DIP switch bit 4 |
| Analog 04 | I | Option bits | DIP switch bit 5 |
| Analog 05 | I | Option bits | DIP switch bit 6 |
| Analog 06 | I | Option bits | DIP switch bit 7 |

Version A above, is for our experiment without a H-Bridge.

Version B, to try with an H-Bridge, connects 6 digital lines D03, D06, D07, D08;  
D09 unconnected; D10, D11, D12, D13 reserved for SPI

## Sequence Diagrams

There is one sequence diagram presented on the next page. This are useful to understand the sequence of interactions between the User Interface, the Server, the ClockWall Controller, the Row Controllers and the Motor Controllers.

### ClockWall Sequence Diagram

Figure : ClockWall Sequence Diagram

# Packet Formats

This section will detail all the packet formats of the messages. There are really four different protocols, so we have to describe each

## User Interface to ClockWall Server

tbd

## ClockWall Server to ClockWall Controller

Tbd

## ClockWall Controller to Row Controller

**SetTime**( *col*, *t* ) -- send to column *col*, a msg to set the millisecond system clock to time *t*

**MotorHighNoon**( col, *t* ) -- send to column *col*, a msg to at time *t*, set the clock hands to 12:00.

**MotorStepCW**( col, *t, nsteps* ) -- send to column *col*, a msg to at time *t*, do *nsteps* clockwise steps

**MotorStepCCW**( col, *t, nsteps* ) -- send to column *col*, a msg to at time *t*, do *nsteps* counter-cw steps

**MotorBrake**( col, *t* ) -- send to column *col*, a msg to at time *t*, do a Short brake

**MotorStop** ( col, *t* ) -- send to column *col*, a msg to at time *t*, do a full Stop

## Row Controller to Motor Controller

**SetTime**( *t* ) -- set the millisecond system clock to time *t*

**MotorHighNoon**( *t* ) -- at time *t*, set the clock hands to 12:00.

**MotorStepCW**( *t, nsteps* ) -- at time *t*, do *nsteps* clockwise steps

**MotorStepCCW**( *t, nsteps* ) -- at time *t*, do *nsteps* counter-clockwise steps

**MotorBrake**( *t* ) -- at time *t*, do a Short brake

**MotorStop** ( *t* ) -- at time *t*, do a full Stop

\_u16 **MotorStatus**() -- return current status, structure tbd

.

# Summary

We have documented the …

## Cost Effectiveness

We need to build 288 Motor Controllers. I believe these could be built for a target price of between $10 and $20.

Then, add 12 Row Controllers. Buying off-the-shelf “MSP432 LauchPad” at $13 each is very inexpensive, and low-risk.

Then, add the ClockWall Controller, an IOT Device. Budget $50 for this.

Each of the 288 Motor Elements need a housing, a bezel and hands.

Finally, the entire thing needs to be mounted on substrate.

The following costed-BOM includes a slight (<10%) unit overage for broken components.

|  |  |  |  |
| --- | --- | --- | --- |
| Count | Item | Estimated cost each | Extended |
| 300 | Motors | $5 | 1500 |
| 300 | Arduino Nano | $5 | 1500 |
| 300 | PCB, with H-bridge and misc. | $5 | 1500 |
| 300 | Motor elements | $3 | 900 |
| 14 | Row Controllers | $13 | 322 |
| 2 | ClockWall Controller | $50 | 100 |
| 1 | ClockWall Substrate | $1000 | 1000 |
|  |  |  | 6822 |

## Robustness

Discussion of the robustness of the design

## Remaining Work.

What remains to be done…