6 Electromechanical Assemblers

As might have been suggested by the difficulty of the work in Chapter 5 designing a single example of a kinematic state machine for programmable assembly, a more general tool for studying the programs & the kinetics was desired. This was deemed sensible rather than designing a specific mechanical implementation for any particular state machine or algorithm that was desirable to test.. To that goal reprogrammable electromechanical 'units' were designed as a physical 'simulator' or 'emulator' for studying the properties of programmably assembling systems. It will be seen that these units also suggest the utility of reprogrammable assembly. I mention here with gratitude the assistance of Dan Goldwater with the development of the electronics in these devices[1].

6.1 Design

Of most interest to me is the study of self-assembling (or programmably assembling) systems where the amount of logic within each component is minimal, in fact so small that the logical operations can be implemented without digital logic. Without this constraint I do not believe these types of components will ever scale down to nm and um components, nor will they be cheap enough to be implemented in sufficiently large number to be interesting. This dictated many of the design choices in building these particular components such that they would emulate similar systems. For example only nearest neighbour communications were implemented to emulate systems where communication between parts can only be achieved by contact, as in for example a kinematic system such as that discussed in chapter 5.

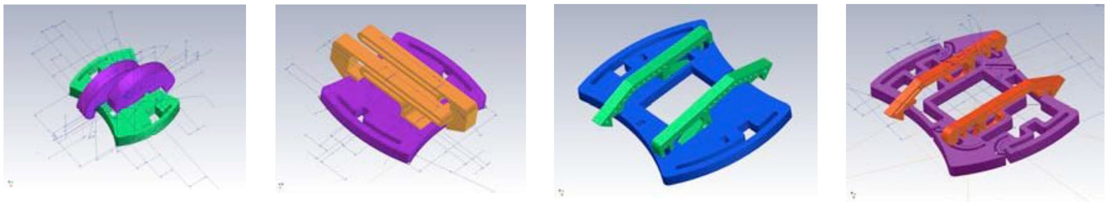


Figure 6-1Early prototypes of magnetically actuated reversible latches.

A tiling had to be chosen for the base of these units. An offset tiling such as that used in the parts in chapter 5 would have been desirable for emulating such a self replicating system, but limiting in studying or building other types of assemblies. Rather a basic square lattice was chosen for it's generality and with the redundant computational cycles of the Electromechanical Assemblers (EMA's) the communication neighbourhood can be increased artificially by message passing. The tiling sits

[1] Dan, as well as being a fabulous friend, knows his way around circuit design, surface mount stuffing, communications protocols, and microprocessor programming. I owe him a debt of gratitude for his assistance in the design, debugging and operation of these units. It had its exciting moments, I hope that can be part of the reward.

on a plane square lattice, but individual tiles are radiused on all edges, 2 convex, 2 concave, as this was shown in early mechanical experiments to increase the likelihood of successful binding upon collision.

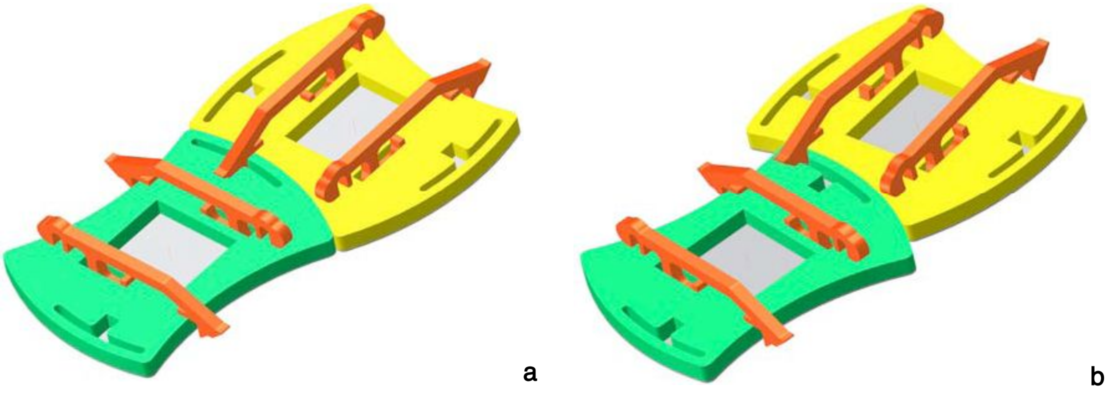


Figure 6-2 a) Two components correctly aligned. The latch has seated in the deeper extension of the latching channel. b) A 'metastable' bond between two components. To increase the chances of successfully binding collisions the latching channel allows a temporary bond that upon further agitation allows the latch arm to sit into the deeper latching position.

To further increase the number of successful binding interactions a 'metastable' binding channel was employed as can be seen in Figure 6-2. Latches can catch in this channel giving an effective approach angle of around 30 degrees. Once caught in the channel the parts are free to settle into a single successful bound position whereupon the communications magnets align. This position can be seen as the deeper cut section in the centre of the binding channel. Figure 6-3 shows the interlocking of 4 successfully bound components. The components in Figure 6-3 are the final design used throughout the experiments in this work.

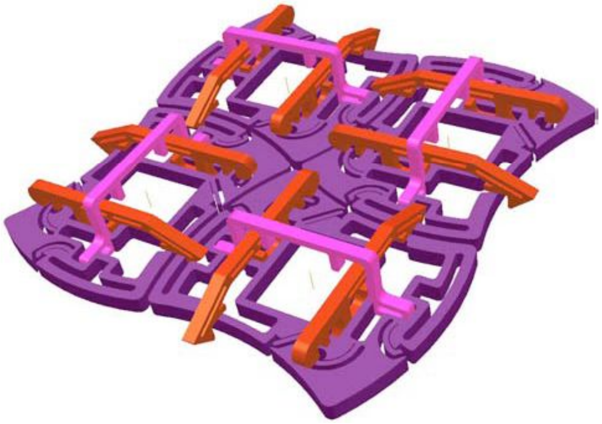


Figure 6-3 Four successfully bound units demonstrating the square lattice that the parts conform to.

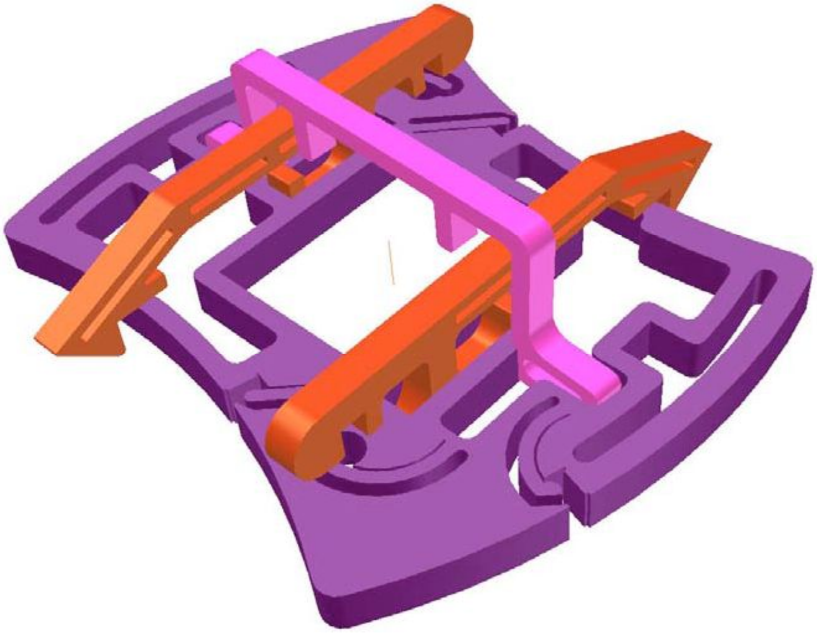


Figure 6-4 Detail of unit base, two latching arms, and a roll cage to prevent arm loss upon heavy collision.

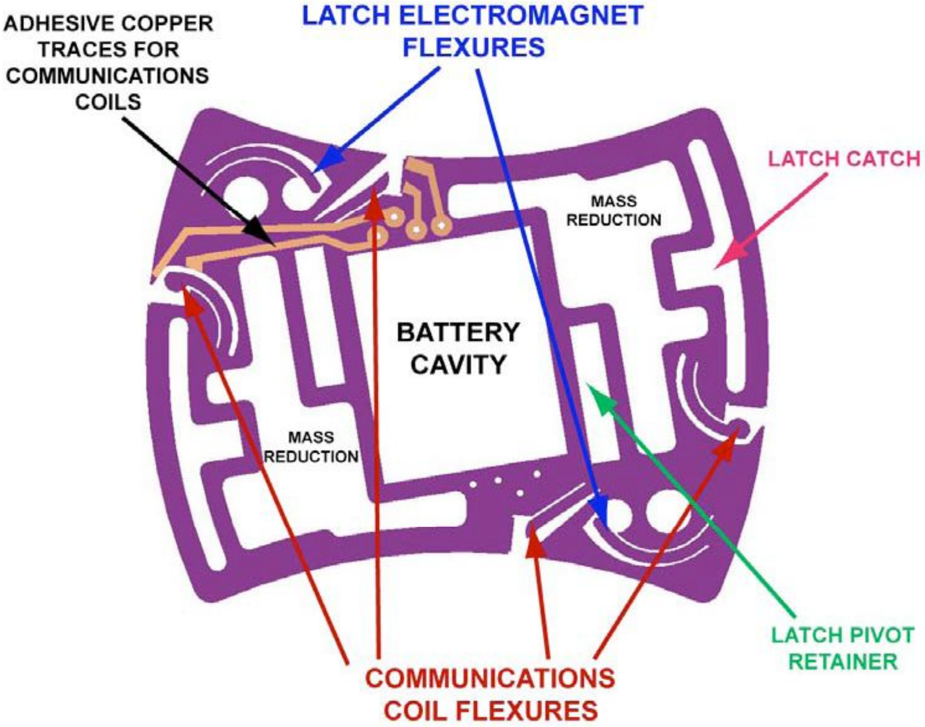

ADHESIVE COPPER TRACES FOR COMMUNICATIONS COILS
LATCH ELECTROMAGNET FLEXURES
LATCH CATCH
MASS REDUCTION
BATTERY CAVITY
MASS REDUCTION
LATCH PIVOT RETAINER
COMMUNICATIONS COIL FLEXURES

Figure 6-5 Projected Detail of unit base.

Figure 6-4 details the acrylic components of each unit, 4 in total. The latches sit in recesses labeled latch pivot retainer in Figure 6-5. The center of mass is on the hook end of the latch pivot such that by default it sits down. The roll cage, or retainer, sits over the two latches to prevent latches from falling out upon heavy collision. Design was done with assembly in mind such that all parts were designed for press fit or seating by gravity without bearings or axles. Figure 6-5 clearly illustrates the flexures which were used for alignment and retainment of both the communication coils and the latching electromagnets. With magnets installed the flexures acted as retaining springs. The battery cavity is also illustrated and was a snug press fit. The circuit board sat atop the battery held in place with double sided tape. The rechargeable lithium batteries with sufficient energy capacity dictated much of the size of the units as nothing smaller than 20x25mm batteries had enough capacity to run the units for the estimated 4 hours required for larger/ more complex assemblies.

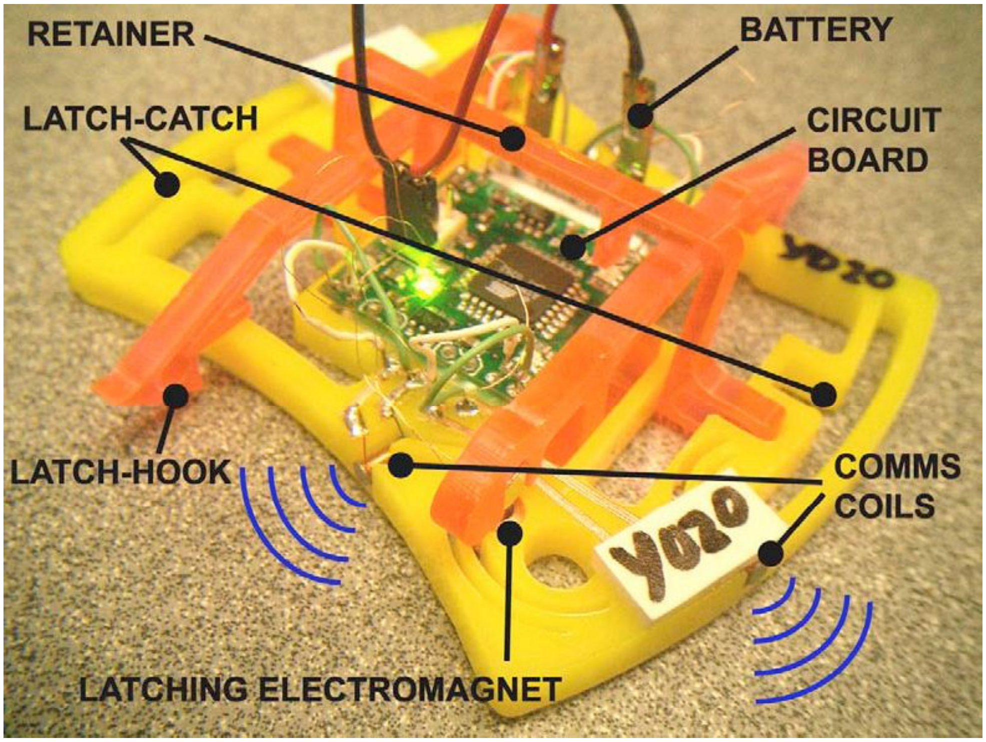


Figure 6-6 Detail of fully assembled unit.

Self adhesive copper traces were cut on a vinyl cutter and adhered between the communications magnets and the circuit board. Wherever possible mass was removed from the units by cutting holes and recesses as can be seen in Figure 6-5 & Figure 6-6. The circuit board was designed at 20x20mm to sit atop the battery. Wires for traces and electromagnets were routed as short as possible to prevent excess wire from being snagged during the random collisions of the assembly environment. Also shown in Figure 6-6 is the unit numbering to aid in debugging, and the removeable battery connector to allow for battery recharging and programming – in retrospect I would have designed for IR or other wireless programming to save the laborious and time consuming process of re-programming each by hand and for a less fragile recharging method – units were often damaged when removing the battery connector.

6.2 Latching

A reversible latching mechanism was required to implement the logic. This latching had to be designed with minimal power consumption in mind. Purely electromagnetic latching was experimentally determined not to be robust enough not to break apart upon random collisions. An electromagnetically activated physical latch was therefore used. A permanent magnet (NdFeB) cantilevered in the latch is suspended over an electromagnet in the base. Current applied to the electromagnet pulls the permanent magnet down, lifting the hooked end of the latch on the opposite side of the fulcrum of the latch. In this manner the default position for all latches is down, but ready to lock, and only upon application of current may components disconnect. The advantage of this is that for virtually all algorithms this is the most energy efficient – the electromagnet is by far the largest current draw of each device. The disadvantage is that because the default state is locking, in the event of a unit failing (power loss, logic loss etc) it cannot release from surrounding components – it is not a failsafe design, but was consciously chosen as such as power was the most limiting factor at this scale.

6.3 Specifications:

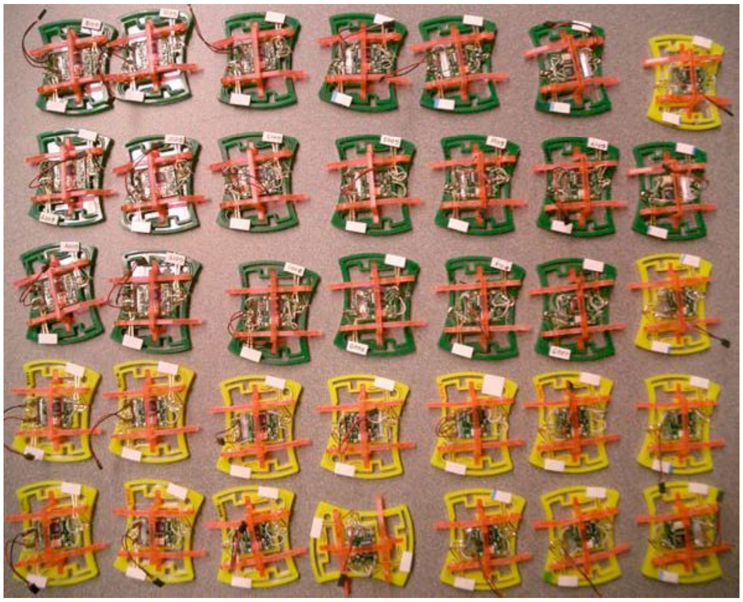


Figure 6-7 Mutliple units ready for experiment. Each is individually numbered.

Electromechanical Autonomous Assemblers: Components.

logic	• Atmel Mega8 microcontroller: 8MHz operation • 10kHz 10bit A/D • 8K program storage + 1K RAM
battery	3.7V, 150mAh Lithium-Polymer rechargeable: 4.2gm
Electronics	A few LED's, FET's and discretes
Latch Actuation	• 2x(4x4x4mm) electromagnets, 350mW. Custom wound, 700 winds, 42 gauge wire. • 2x(3x3x3mm) rare-earth permanent magnets. NdFeB, www.amazingmagnets.com
Communications	• 4 Full duplex concurrent channels • Wireless, software adjustable, 1 – 10mm range • Send and receive directly with microcontroller • Small size: 2x2x4.5mm electromagnets, 2-wire connection • Low power: 2mW transmit, passive reception
Base	Corners sit on 50x50mm square. Laser cut 4mm acrylic with features (6.9gm) over .9mm acrylic base (4.7gm). • 4 x flexures for squeeze fit of comms magnets • 2 x flexures for squeeze fit of latch magnet • Total mass: 11.6gm
Latches	Laser cut 3.2mm acrylic Press fit permanent magnet hold
Roll-cage (prevent latch bounce-out	Laser cut 3mm acrylic Press fit to base
Electronics package (logic battery actuators)	Total size: 25x20x6mm, 6g
Total package	Size: 50x50x15mm, ~26gm.

Table 6-1 Specifications for electromechanical units.

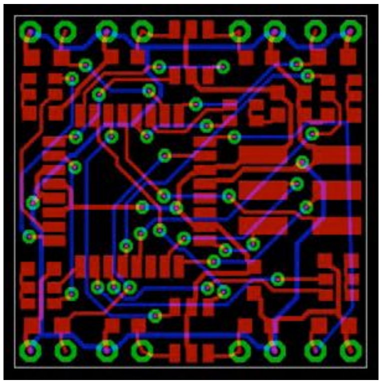


Figure 6-8. Circuit board layout. Circuit board is 20mm on an edge.[2]

[2] Design and implementation of circuit board with assistance from Dan Goldwater.

6.4 Debugging and maintenance.

50-100 of anything is probably the worst number to make. Below 10 hand assembly and debugging is simple. Above 1000 outsourcing the manufacturing is economic. Unfortunately this project sat in the middle. As such tools for debugging and maintenance were also built to simplify the process of keeping sufficient units running correctly.

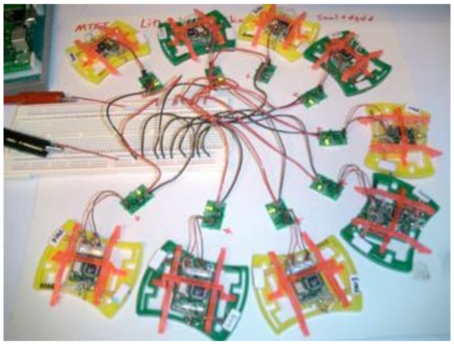


Figure 6-9 Multi-unit recharging bay with a simple circuit board driving each.

Figure 6-9 illustrates a multi-unit recharging bay, each battery served by a simple charging circuit board specifically designed for the Lithium Polymer batteries.

Figure 6-10 shows the communications testing unit. The communications coils are tested by placing a unit in the centre or this piece. LED's on the circuit board at left in this photo indicate whether each communications coil is capable of both send and receive.

Figure 6-11 shows 4 units each used in debugging and programming. Units labeled D1 and D2, at left, were designed to probe the internal state of other units, and to provide signal-level breakouts for probing with an oscilloscope. The two units on the right in figure Figure 6-11 are for reprogramming the units without needing to connect them to an external computer. There was enough space in memory on each unit to store 4 programs at any given time as well as a colour identifier. The top right unit would program any given unit to be either green, or yellow for algorithms where colour was sensitive, and the lower right unit would switch the units between 4 programs (numbered 1-4).

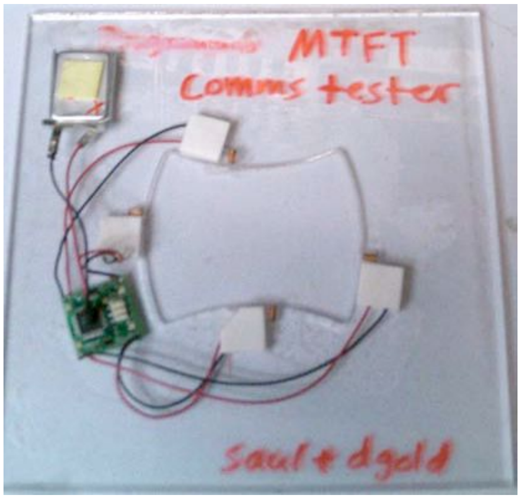


Figure 6-10 Communications testing unit. This unit tests the communication coils on all 4 sides of a unit placed in the centre.

Further debugging functions were programmed into the units such as an auto latch test upon powering up (toggle both latches up and down) and state indicator sequences in the LED's on each circuit board. Similarly each unit would identify whether it was transmitting or receiving at any given time via two colour (red/green) LED's for each of the 4 communications coils. A similar LED indicator was associated with each latch such that faulty latches could be identified during experiments. Low battery charge, for example, could prevent latches lifting during experiments, similarly in the occasionally violent environment of a random-collision air table, electromagnet wires would occasionally fail.

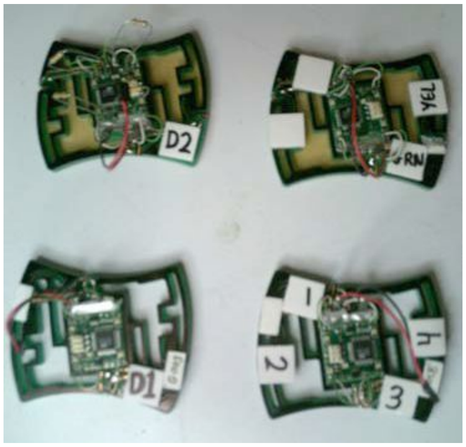


Figure 6-11 Debugging and reprogramming units.

6.5 Communications

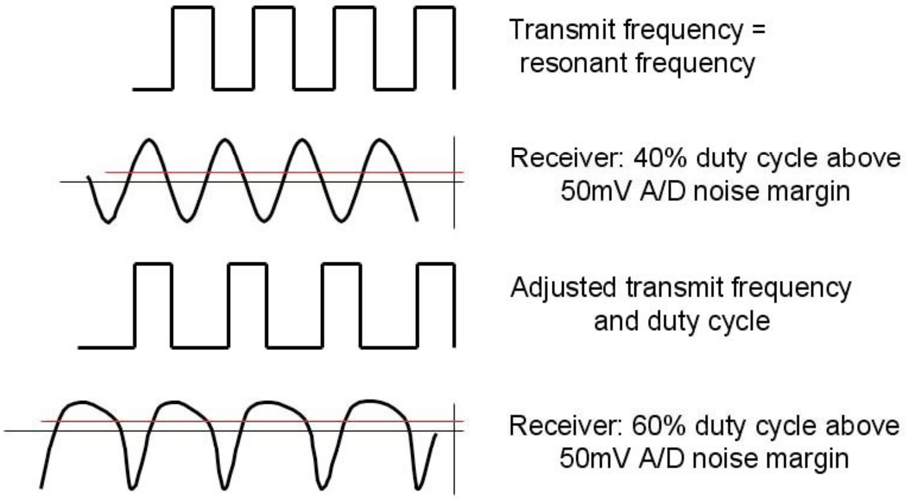

Transmit frequency = resonant frequency
Receiver: 40% duty cycle above 50mV A/D noise margin
Adjusted transmit frequency and duty cycle
Receiver: 60% duty cycle above 50mV A/D noise margin

Figure 6-12 Schematic of transmit and receive implementation in communications channel.

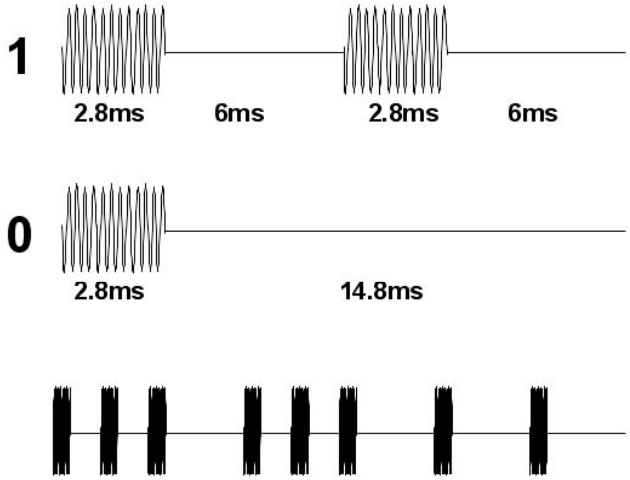


Figure 6-13 Schematic of collision avoidance implementation on communication channels. Coils receive when not transmitting & randomize transmit to avoid collisions. Packets are 12 bits (7 data bits) 210ms transmit time.

Figure 6-12 & Figure 6-13 [3]demonstrate the method by which communications was achieved via the coils on each of the 4 sides of the units. Whenever the coils were not transmitting their state they

[3] The communications protocols shown here were designed and implemented with the assistance of Dan Goldwater.

were listening for incoming tiles. The transmit times were randomized to avoid collisions. In such a communications environment many error sources exist: crosstalk, transmit collisions and communication distance variation. Multiple methods of error detection were implemented to ensure high communications integrity they can be simply described as:

- Packet size: discrete 12 bit packet must be received
- Fixed bits: 4 bits of the data packet are constant, these are spread through the data packet
- Parity bit: parity bit is appended to the data packet
- Redundant transmission: two identical transmissions must be received in sequence to be considered valid

Experimentally it appeared that this communications protocol was sufficient for the scale of these experiments (both temporal and spatial)

6.6 Assembly Environment

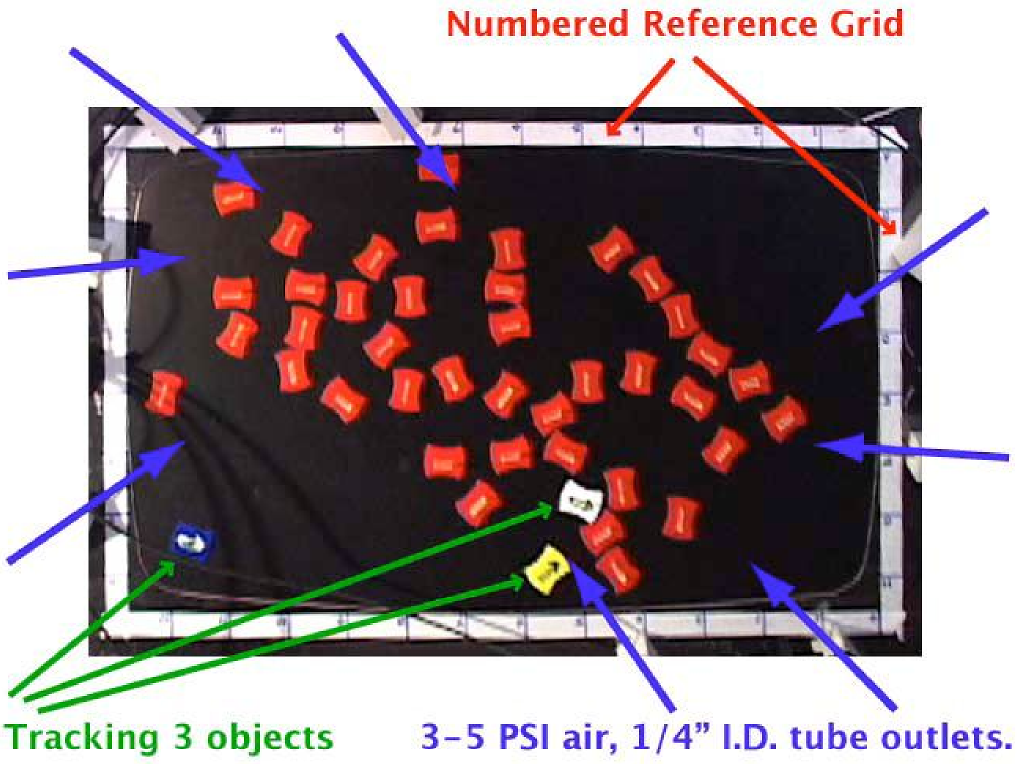


Figure 6-14 Experimental verification of system dynamics with dummy parts. The direction of incoming air streams is indicated by the blue arrows. Three uniquely coloured dummy tiles were tracked on video.

Figure 6-14 illustrates the assembly environment. A custom-built air-hockey table was used to constrain assemblies to 2 dimensions and simulate a frictionless environment. 0.8mm holes were drilled throughout the air-table at 1inch intervals on a square grid. A 130 x 75mm square constraining frame was cut in a 3mm polycarbonate sheet that sat over the air table. The lip of this polycarbonate sheet acted as a retaining barrier constraining the parts within those bounds. The maximum floating weight for a tile of the given geometry was determined and units were designed and lightened to be under this weight. This maximum weight was 30+/- 2 grams.

To introduce randomness to the system 8 air jets were introduced from the sides of the table. These air jets were run through 1/4" I.D. polypropylene tubing that was anchored by weights at the sides of the table. Unregulated house-air was run through the tubes and a valve and pressure gauge was used to keep the exit pressure between 3.5 and 4 PSI. Natural variation in the house-air pressure meant occasional variations above and below these values during the course of lengthy experiments. These values were experimentally determined to maximize reaction speed and minimize random unit damage due to high energy collisions. Qualitative observation of collisions in this environment gave confidence in them being largely elastic collisions.

6.6.1 Data recording

All experiments were recorded with a Sony miniDV camcorder equipped with time stamp. Video was processed and analysed with a combination of I-Movie, Final Cut Pro, Adobe After Effects, and Quicktime Professional.

6.7 System Dynamics

System dynamics were experimentally verified using the set-up in Figure 6-14. Fourty-two dummy units of equivalent mass and 'sail' area (the vertical area presented to the side air jets) were run on the table for 5 minutes at a time. 3 different coloured (blue, yellow, white – marked with a direction vector) dummy tiles (similar in mass and sail area) were tracked. At 1 second intervals during the course of the experiment the X and Y coordinates and orientation (divided into 22.5 degree slices of the 360 degrees of rotation) were recorded.

Figure 6-15 shows screenshots at 15s intervals during one run of the control experiments. The orientation of the parts and their position is clearly visible.

Fifty seconds worth of the data is shown in Figure 6-16. More data points did not illuminate anything extra and only confused the graph. As might be expected the components tended to clump in the centre of the board at furthest distance from the incoming 'randomising jets'. Overall system dynamics could be observed qualitatively with trends in clumping for different positions and orientations of the incoming jets.

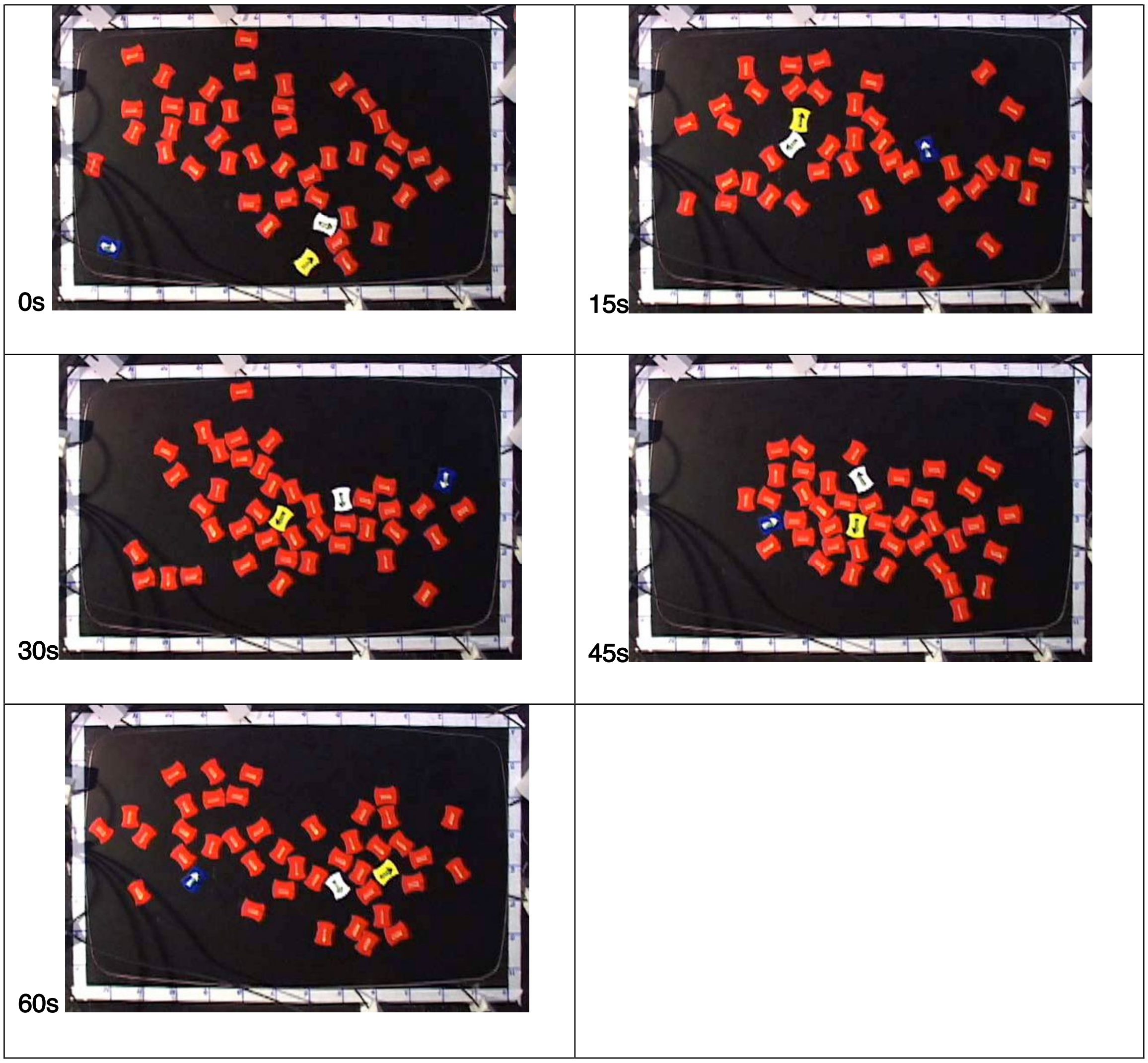


Figure 6-15 Screenshots from system dynamics control experiment taken at 15 s intervals. Position and rotational orientation are recorded for the blue, yellow, and white tiles.

A set of orientations for all 8 jets that was least violent in terms of high speed individual collisions, yet maximized number of collisions was decided upon qualitatively. The data in Figure 6-16 was taken with those positions and orientations which are illustrated as blue arrows in Figure 6-14. The jet positions were retained for all experiments. This dynamics data convinced the author of sufficient local randomness as seen by each tile to assume the entire system provided random collisions over the typically 30-60 minute time intervals of the programmed tile experiments.

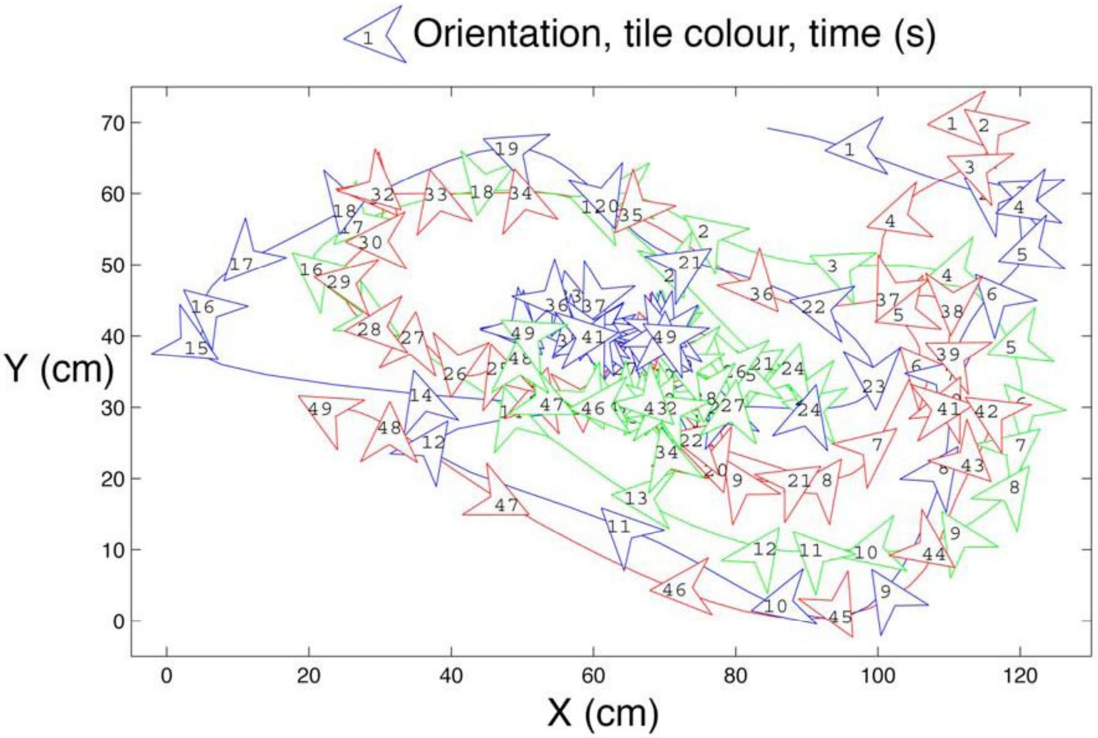


Figure 6-16. Graphed data for a random 50 second interval in dynamics control experiment. Three tiles are tracked at 1s intervals. Data is shown as an arrow indicating orientation, within which is a time stamp (seconds) and a loosely fitted trajectory line. Blue tile shown as blue, Yellow tile as green, White tile as red.[4]

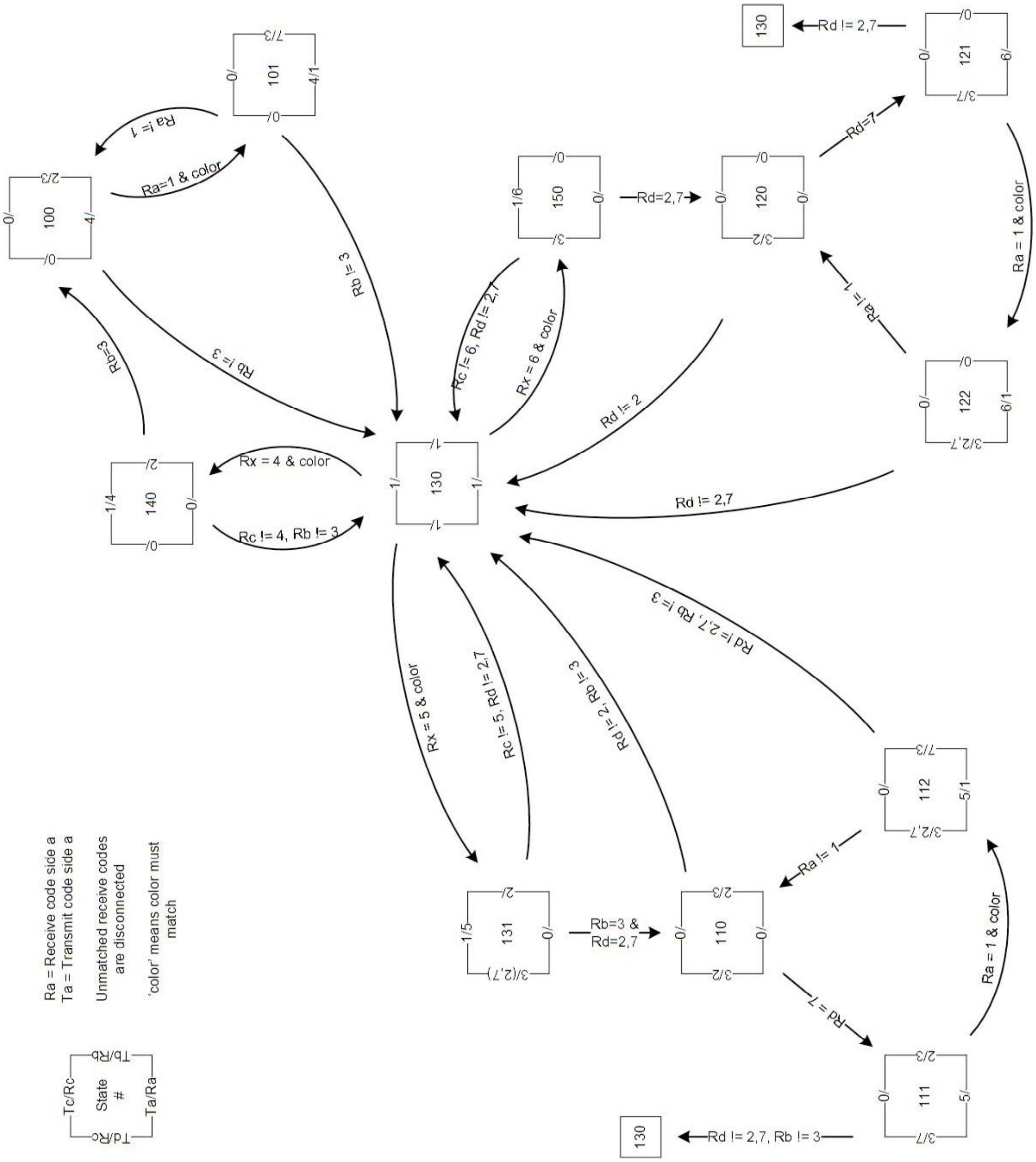
6.8 Programming

The state machines were implemented in C code compiled for the Atmel chip. The code will not be presented here as the details of any particular implementation are not important to the thrust of this work. Rather the programming model for implementation of the finite state machine active on each tile shall be presented. The state machines were implemented as schematically outlined in Figure 6-17. Transmitted and received codes (including unit color) are represented for each of the 4 faces. The received code and color-match that result in a state transition are labeled alongside the state transition arrow which shows the transition from one tile state to another.

Because of the square lattice tiling rather than the offset square lattice tiling used in chapter 5, more than the 6 states required for the mechanical replicating state machines were used. These were essentially memory states such that data could be communicated to non-contact tiles. Extra states

[4] Matlab generation of these dynamics with kind assistance from James McBride.

were also used to handle the time delays in communications that are not seen in the kinematic model presented in Chapter 5.



Ra = Receive code side a
Ta = Transmit code side a

Unmatched receive codes are disconnected

'color' means color must match

Figure 6-17. Schematic of programming model for implementing finite state machines on individual units. The most complex of the state machines, the self-replicating code, is presented in this diagram. Diagram with assistance of Dan Goldwater.

6.9 Experimental

3 principal algorithms were run with the electromechanical units. Line formation, error-preventing crystal formation, bit-string replication, and logic-less control crystal formation. There were two principal tile colors, green and yellow, with a few specially appended colours to denote (red) seed tile for crystalisation and (black) infectious tile for reprogramming virally. Color dependent, and color independent experiments were run for both line formation and error-preventing crystal formation. In color dependent experiments the extra state of tile color was considered in the state machine giving single colored lines (line formation) or chess board patterns (error-preventing crystal formation). In colour independent experiments tile color was ignored giving multicolored lines or randomly colored crystals. System dynamics for all experiments were controlled according to section 6.7.

The state diagrams are presented slightly differently here to more accurately reflect their implementation in this system. Rather than represent the state of the entire tile, the state of each face is shown. This state describes the broadcast and receive signals and binding condition for each face.

6.9.1 Logic-less control crystal.

This is the control experiment which demonstrates the value of logic in self assembly. Obviously these are non-annealing structures as the bonds are irreversible, similar to the glasses described by whitesides in chapter 2. The experiment was run multiple times by not connecting the batteries, thus all latches bound and did not release. The state diagram can hence be drawn very simply as in Figure 6-18. The results from two runs of this experiment are shown in the still frame excerpts of Figure 6-19 and Figure 6-20. The final frame (bottom-most at left in both series) shows the final state of the system at experiment’s end. The two structures are both notably different, as one would expect, and will likely be different for every assembly. Both structures show growth errors where vacancies are entrapped by growth faces. As might be expected these experiments had the fastest kinetics of all assemblies as no time-delays for logic to occur were present, nor searching for colour matches, nor waiting for parts to bind to specific growth faces. These kinetics can be seen most vividly as compared to logic limited crystallization in Figure 6-28.

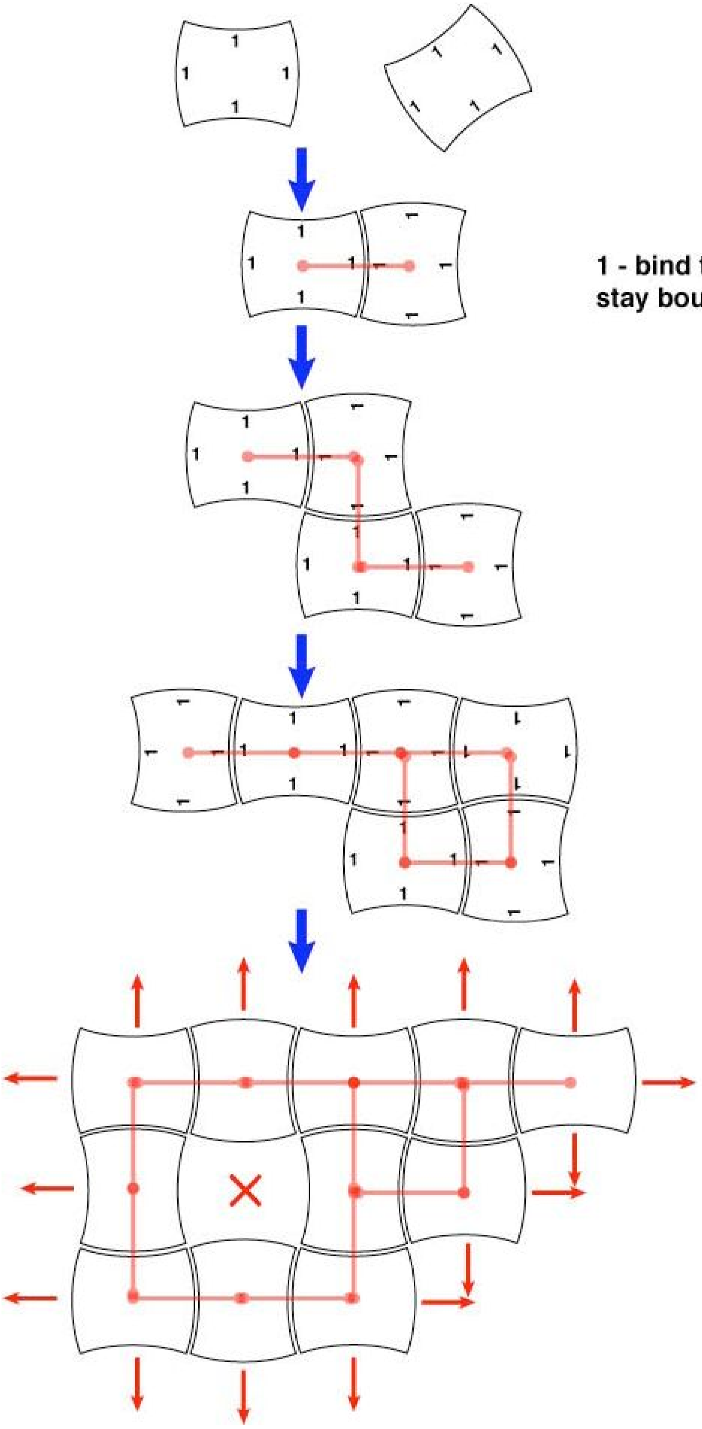


Figure 6-18 Schematic of state machine implementation for logical epitaxy of error free crystals.

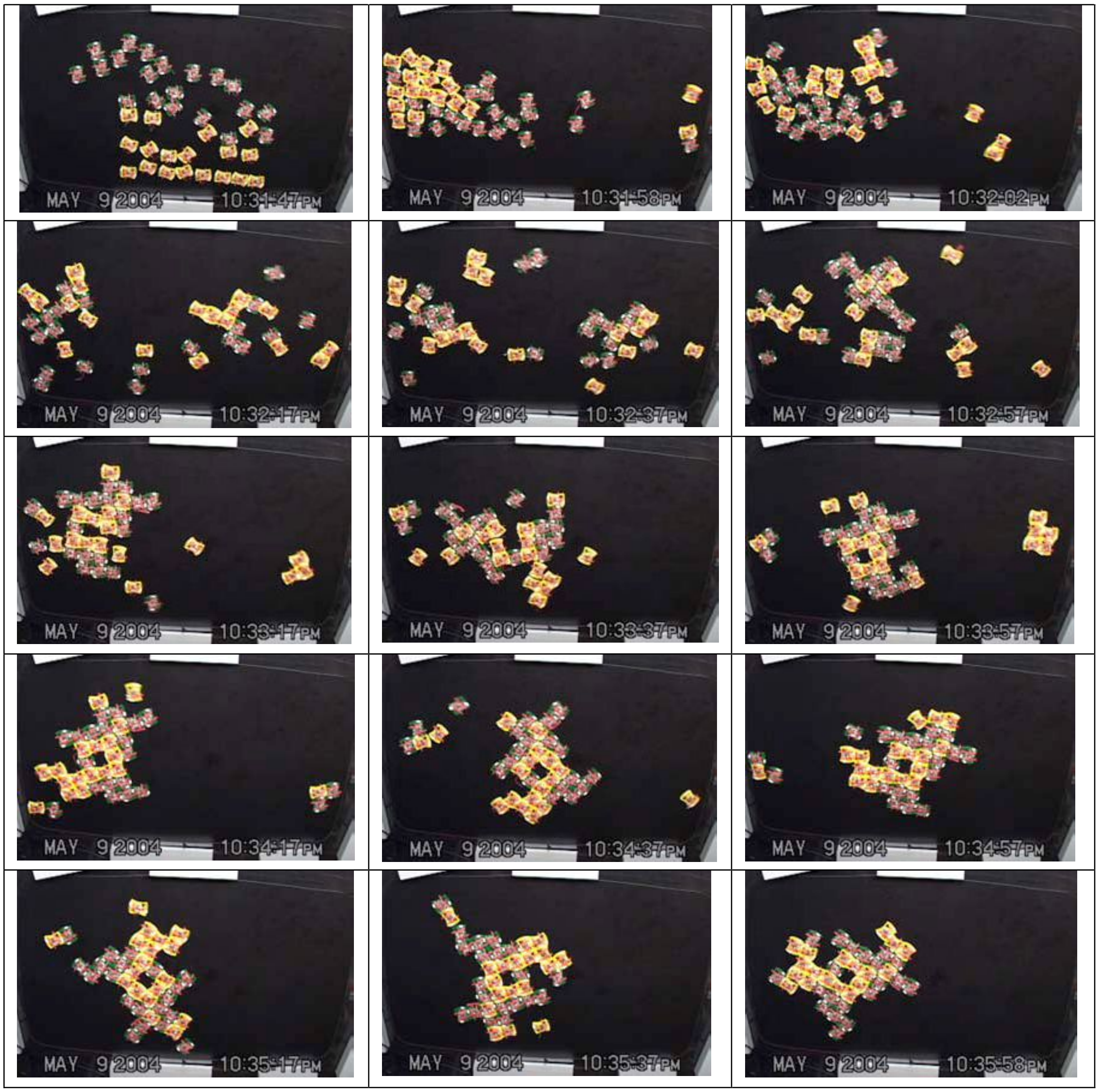


Figure 6-19 Logicless crystal control 1. Series proceeds from left to right, top to bottom. The final state is seen in the bottom-most right hand picture. All units are bound.

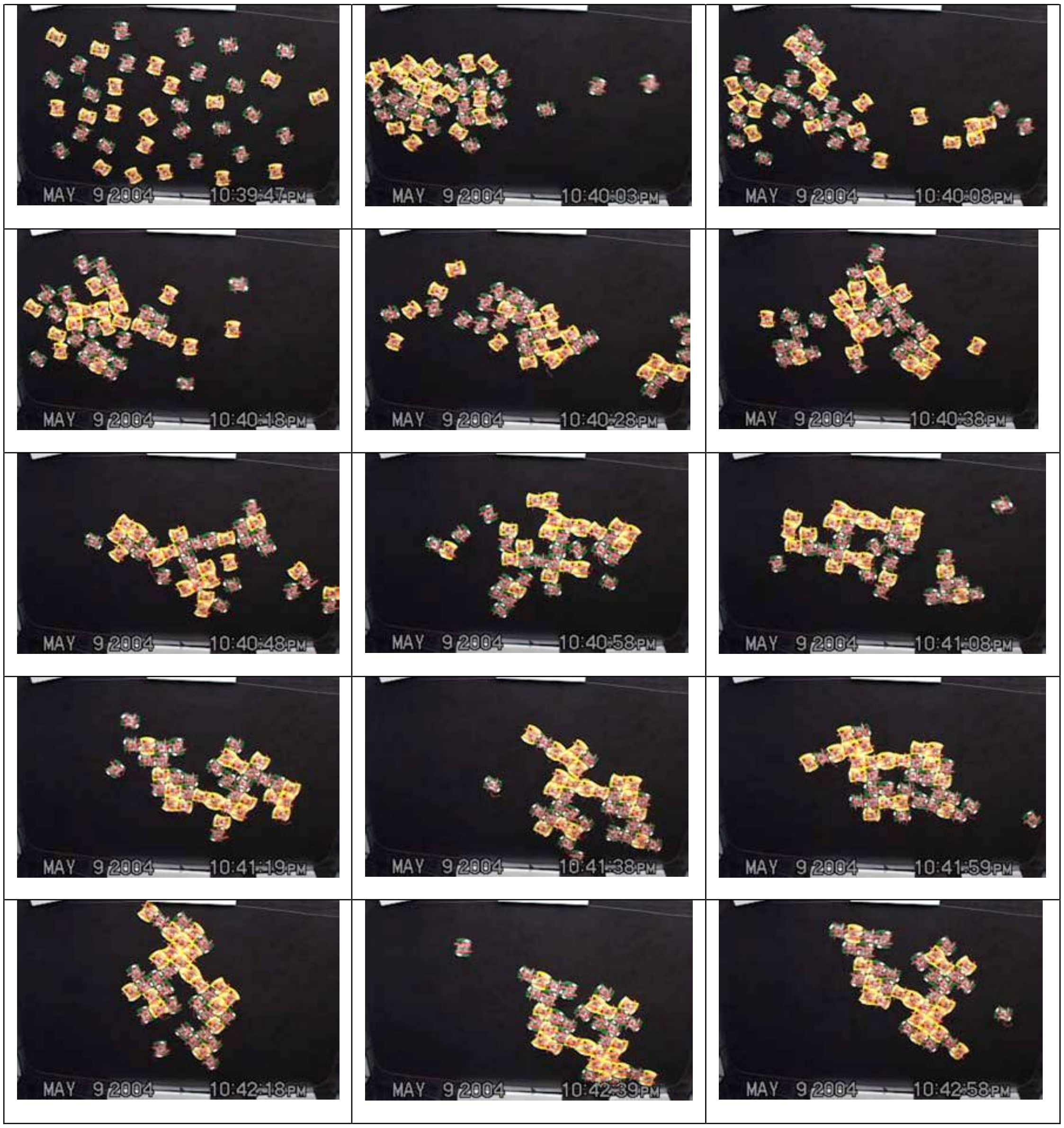


Figure 6-20. Logic-less crystal control 2. Series proceeds from left to right, top to bottom. The final state is seen in the bottom-most right hand picture. All but one units are bound.

6.9.2 Line formation

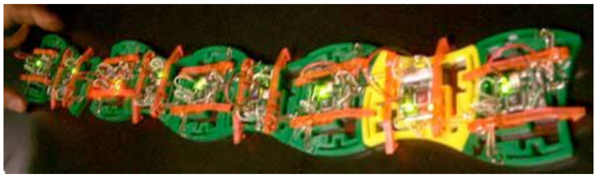


Figure 6-21 Lines are linear, end-end, string formations of tiles, either color specific or mixed (as is shown above).

The simplest program to be implemented was a polymerization, or line forming algorithm. This algorithm produces linear strings such as that in Figure 6-21. This algorithm was tested both for lines of a single color and lines of mixed colors. The line formation state diagram is illustrated in Figure 6-22. Screenshots from the implementation of this algorithm for color specific lines can be seen in Figure 6-23. The difficulty of implementing this algorithm on the experimental set-up as described is that the small ‘reaction frame’ defined by the polycarbonate fence limited quickly the lengh that the polymer could grow to, and indeed it would quickly jam between sides of the frame, occasionally with enough force to break the bonds.

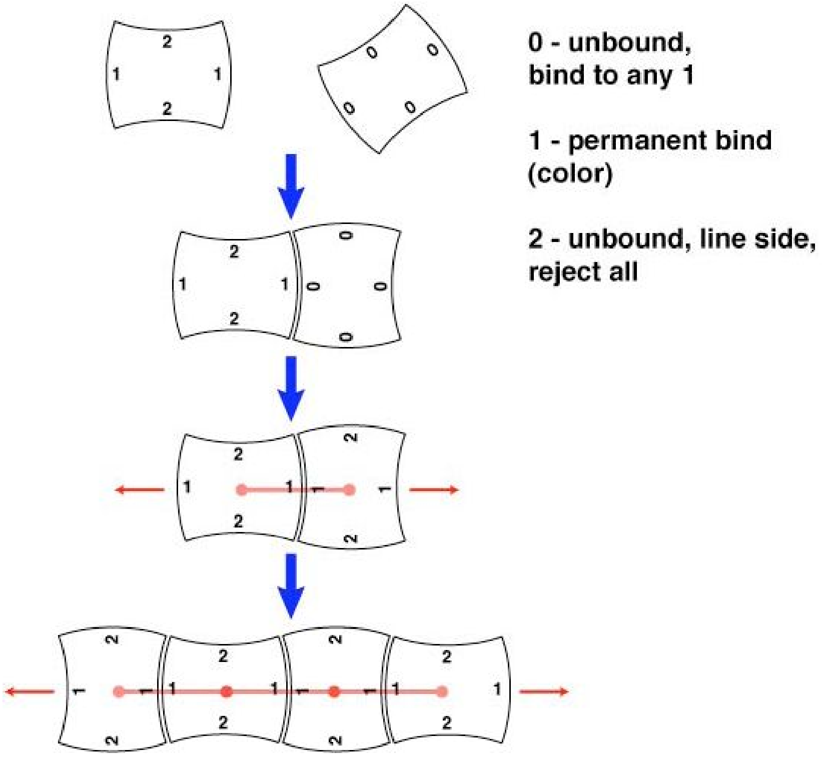


Figure 6-22 Schematic of state machine implementation for polymerization or line formation.

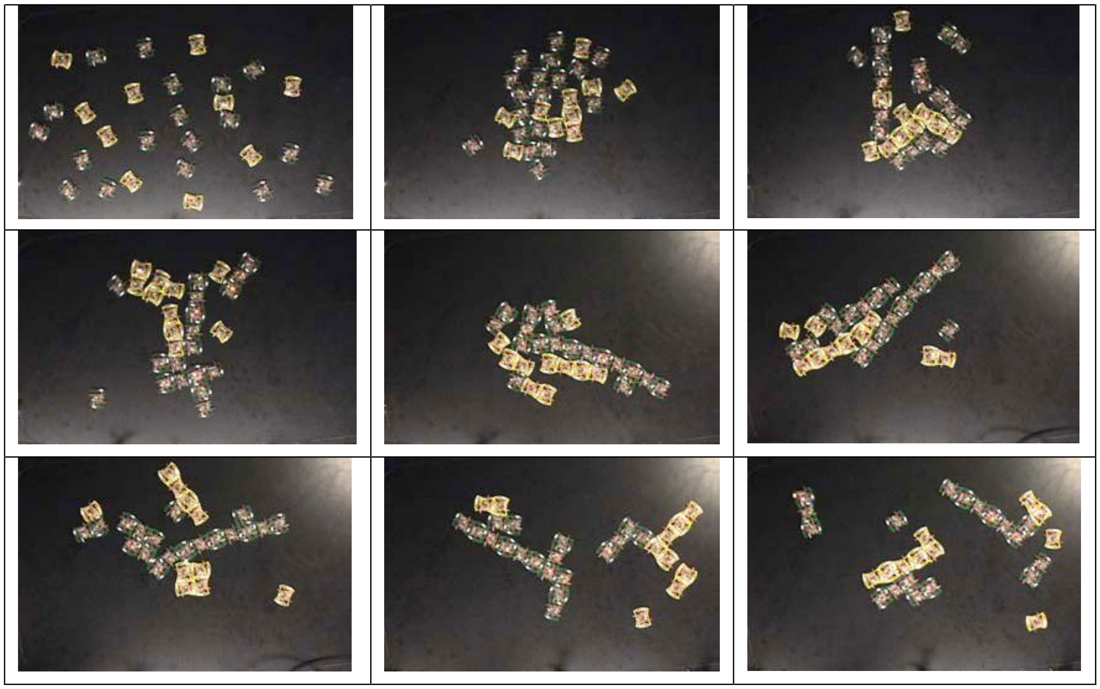


Figure 6-23 Screenshots from a line formation experiment. Series proceeds from left to right, top to bottom. The final state is seen in the bottom-most right hand picture. A number of different polymer chains can be seen to have grown.

6.9.3 Error-preventing Crystal Growth.

I will also call the error-preventing crystal growth algorithm the logical epitaxy algorithm. A non colour specific example may be seen in Figure 6-24. Figure 25 is the non color specific example. Similarly this algorithm could be implemented to produce checkerboard patterns as will be seen in Figure 6-26.

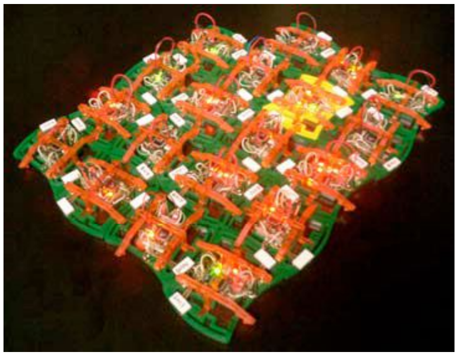
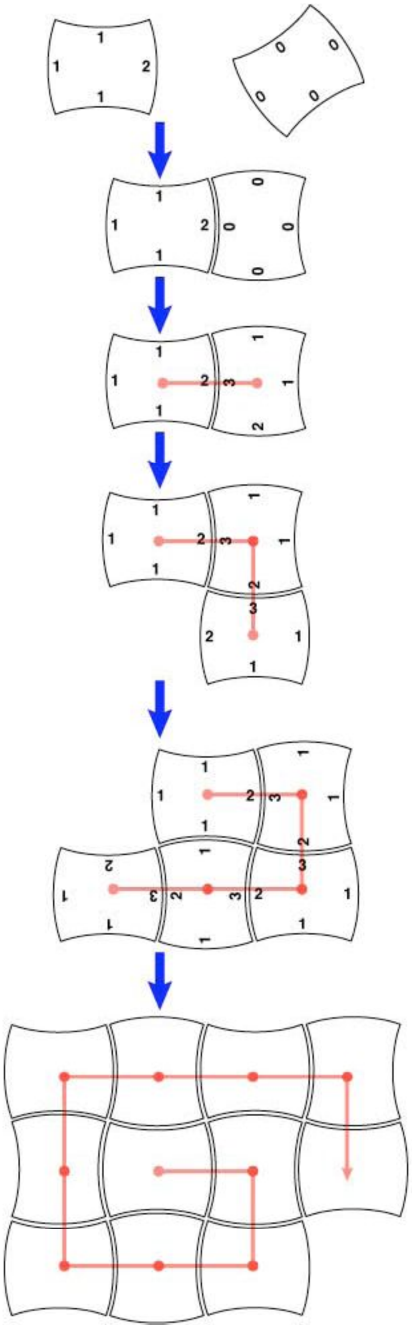


Figure 6-24 Error preventing crystal growth by spiral growth face. Also could be described as logical epitaxy.

The algorithm for logical epitaxy is outlined in Figure 6-25. The crystal can be seen to grow from a spirally defined growth face that only allows a single unit growth face. Screenshots from this experiment can be seen in Figure 6-26, and the kinetics are analysed in comparison to the kinetics of the crystal control experiment in Figure 6-28. It can be seen that the logical overhead of this algorithm makes the growth kinetics significantly slower than random aggregation due to the single, as opposed to many, growth front.



0 - unbound, don't bind unless receive 2

1 - bound/unbound, part of crystal

2 - bound/unbound, growth face

3 - bound message

Figure 6-25 Logic Limited Aggregation algorithm.

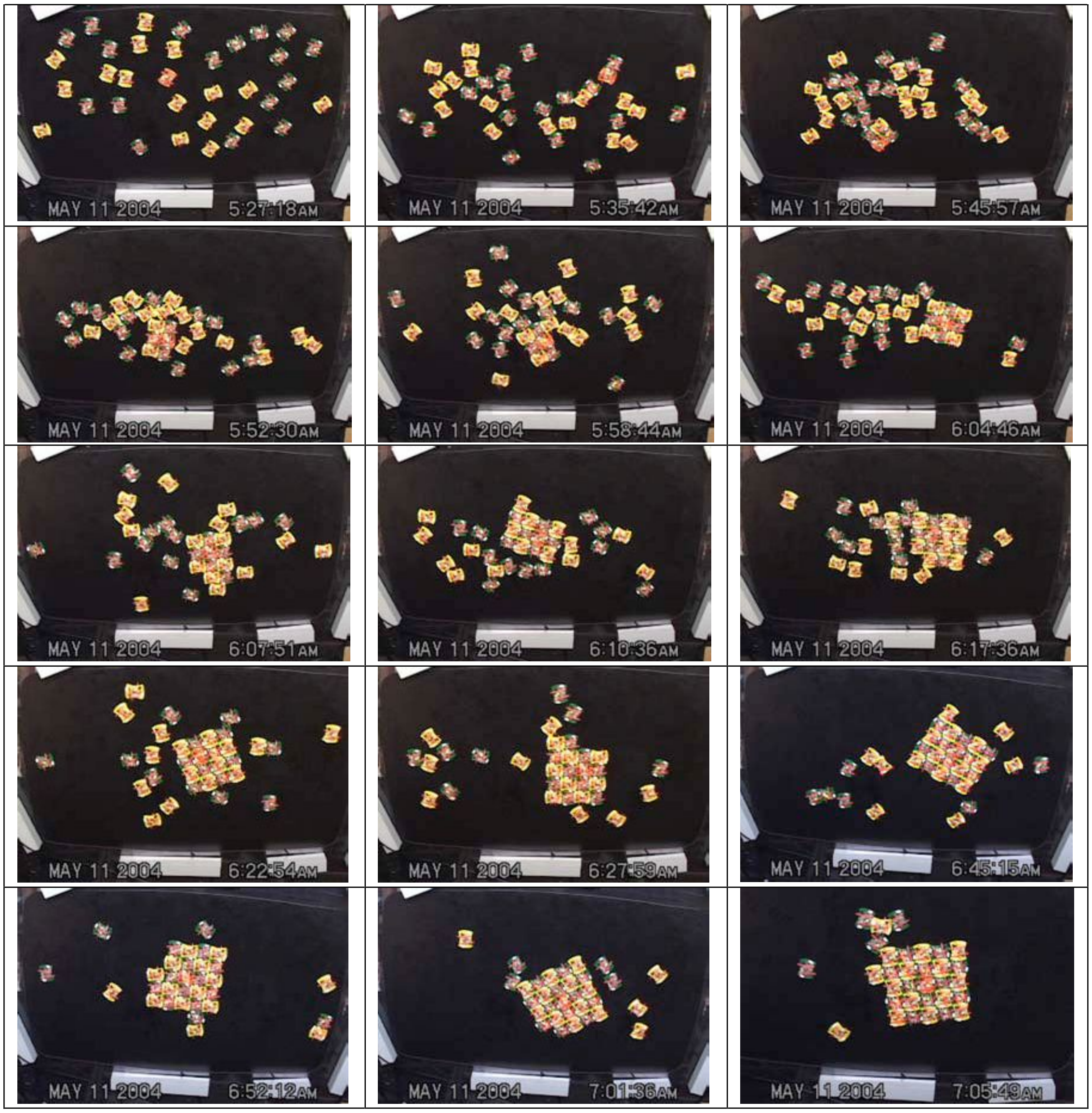


Figure 6-26. Screenshots from a logic limited aggregation experiment. Series proceeds from left to right, top to bottom. The final state is seen in the bottom-most right hand picture.

6.9.4 An autonomous, self replicating machine.

If we define a self replicating machine as a collection of sub-components, capable of replicating itself from a sea of similar sub-components, then the following experiment successfully demonstrates a self-replicating machine. The algorithm for this experiment implements the state diagram of figure 5-18 that was designed for the mechanical system of chapter 5. The implementation of this algorithm was not amenable to the face-state treatment of Figure 6-18 & Figure 6-22. The implementation was done according to Figure 6-17. A length 5 colored polymer, sequence Green, Green, Yellow, Yellow, Green, was introduced into the assembly environment as shown in the 1st of the screenshots of Figure 6-27, all other units were unbound. From this initial condition the experiment was allowed to proceed until 3 copies had replicated. To aid in the visualization of the experiment the 5-mer's were tracked by software (Adobe After Effects) by using contrast tracking of the beginning and end tiles of the 5-mer, from the video track. At the conclusion of the video in the last screen shot the 4 5-mers, 3 replicated and original, can be seen as well as the remaining individual units, one of which is being pointed to. Each replicant is numbered in order of production. Similar experiments were run for 3 and 4-mers.

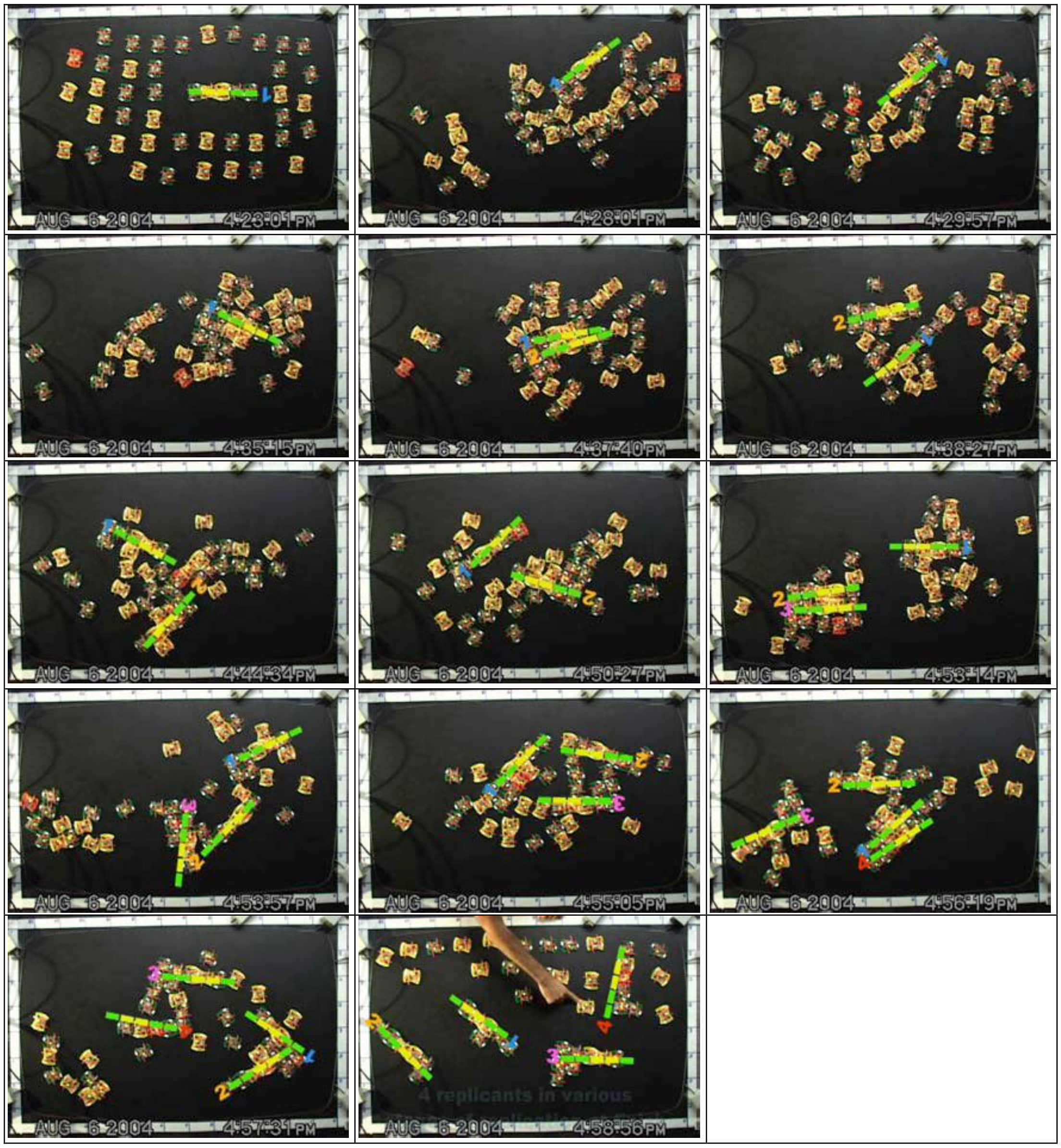


Figure 6-27 Screenshots from a self-replicating bit-string experiment.

6.9.5 Reaction kinetics.

Data collection for the kinetics of these assemblies (also referred to as reactions) is laborious. Video data is analysed manually. The number of bound tiles and other information is recorded along with the video time stamp at periods throughout the assembly by observation of still frames of the video. Some small error is assumed for the counting of bound tiles as video resolution is not high enough to always see whether each tile is firmly bound or about to be rejected. The small number of units in the system also limits the data collection. For example, in the replication experiments where one would expect the process to consume parts exponentially faster with time, the reactants are mostly consumed after a small number of replicants. For all experiments a similar number of units (around 40) were used, although almost 60 units were made, at any given time there were a number being recharged, repaired, reprogrammed, or debugged.

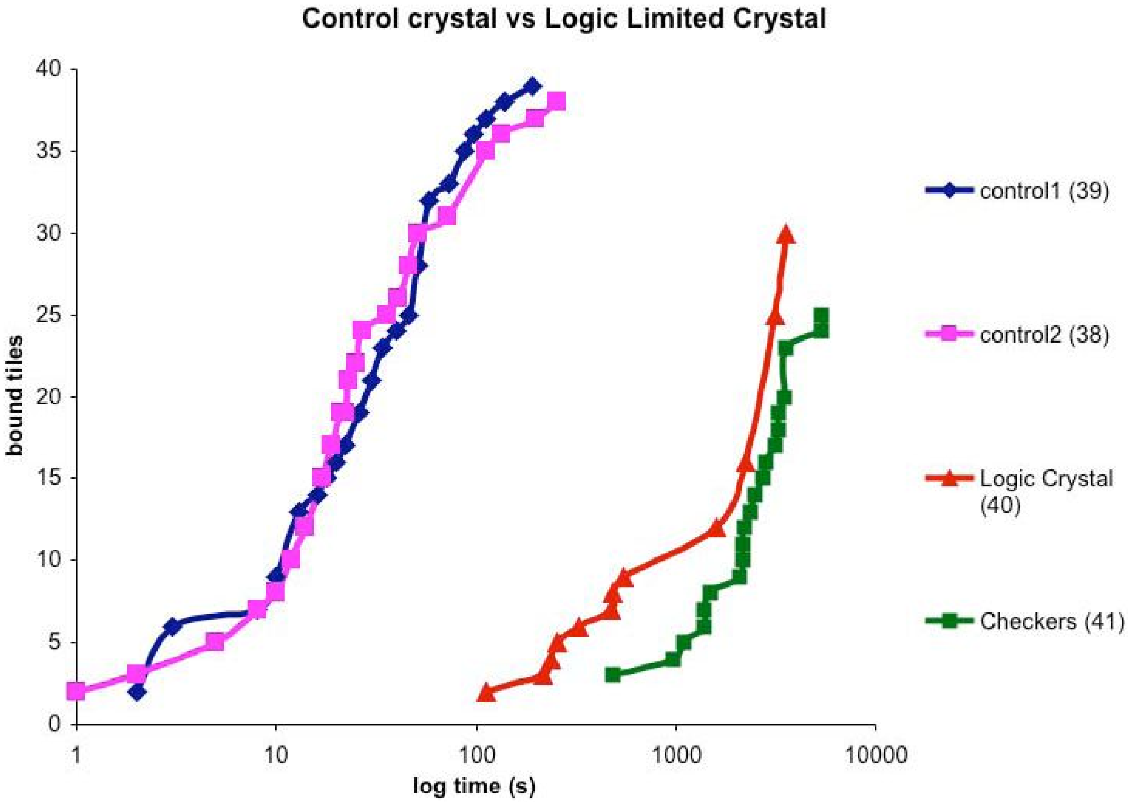


Figure 6-28 Comparing the kinetics of logicless (and error ridden) logicless, irreversible binding, to logic limited aggregation on a spiraling growth face which grows without vacancies or errors.

Figure 6-28 compares the kinetics of the logic limited crystal to the random control crystal. Both the color dependent and color independent data is shown for the logic limited crystal. As might be expected the logicless crystal follows roughly sigmoidal kinetics as might be expected in a small reaction vessel. Slow start with few groth faces, followed by a fast growth period that slows as the reactants are consumed ultimately to stop growth when there are no remaining unbound units. The logic limited crystals might be expected to be basically linear, trailing off as the reactants are consumed as there is always only a single growth face accepting only a single unit. The slow start here is due to the mechanics of the components. The small crystal at first has low mass and therefore low momentum and the kinematic collisions are more successful between heavier components as the latches are more likely to be lifted from their neutral state. As the crystal grows it's mass is higher and the kinetics start to become linear before trailing off as expected.

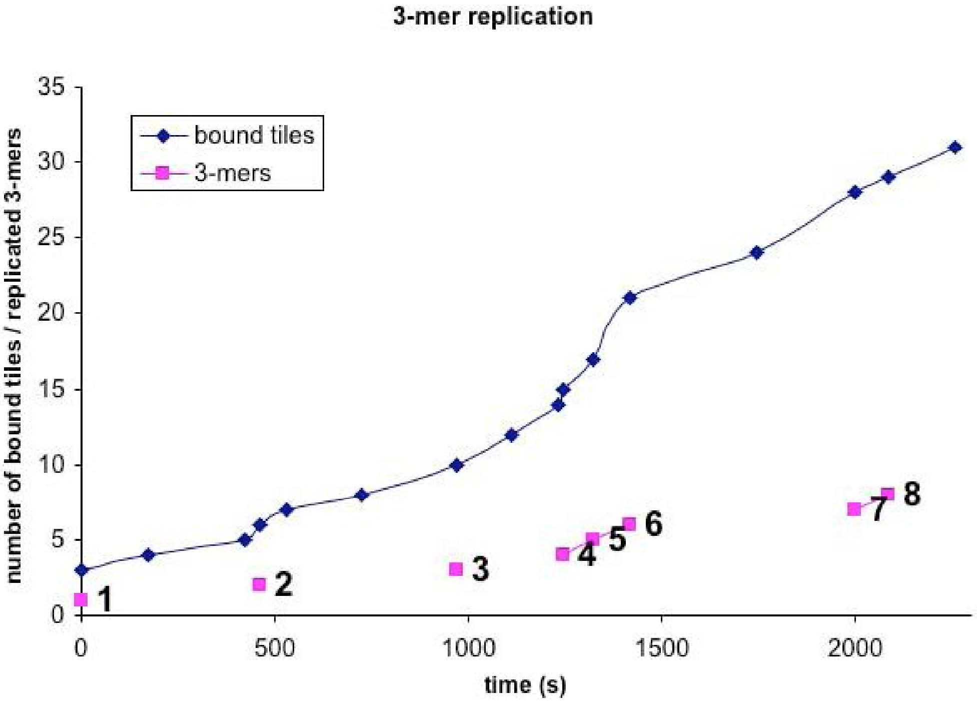


Figure 6-29 Number of bound tiles with time, and number of replicants for a 3-mer replication experiment. Bound individual units shown in blue, complete, bound replicants and original string numbered in pink.

Figure 6-29 looks at the replication of 3-mer's. One should expect exponential reaction rates as the number of growth faces increases exponentially with the number of copies. Unfortunately the tragedy of small component count systems is demonstrated here. The kinetics seems to begin going exponential at the 4th, 5th, and 6th copies, however at this stage roughly half or more of the units are consumed in replicants or partial replicants. The reaction slows. Figure 6-30 shows a similar trend. At the 3rd and 4th copies the bind rate seems to begin increasing, but again, at this stage too many unbound units are consumed. It will be exciting indeed to miniaturise similar components and test for the kinetics in a less bounded reaction vessel by higher reactant number.

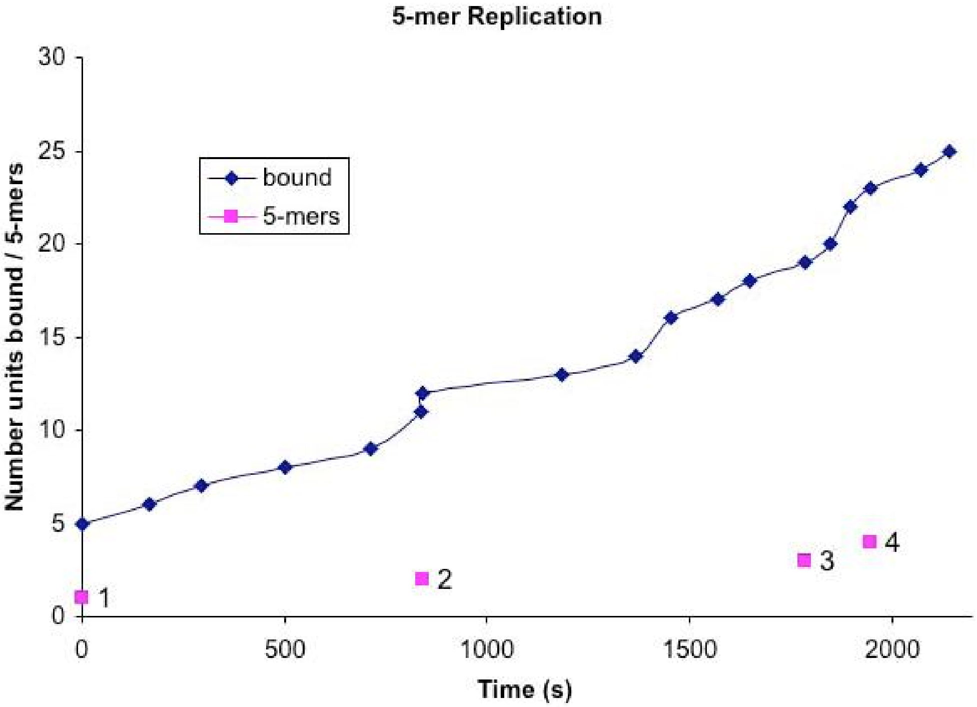


Figure 6-30 Replication of 5-mers. Bound individual units shown in blue, complete, bound replicants and original string numbered in pink.

The reaction rates for all experiments are plotted together in Figure 6-31 (log time) and Figure 6-32. The time cost of logic in the kinetics is clearly shown. The fewer growth faces dictated by the algorithm the slower the reactions. Extra search time is clearly indicated by the fact that color selective algorithms are almost exactly twice as slow as color independent algorithms. This is seen most dramatically in the logic crystal kinetics compared to the kinetics of the checkerboard crystal growth. It would similarly be interesting to run these algorithms in much larger reaction vessels where the stoichiometry of the colored tiles could be varied in color selective algorithm experiments. This particular aspect of this work is highly amenable to computer simulation, and I hope this work informs and to a degree calibrates that type of work.

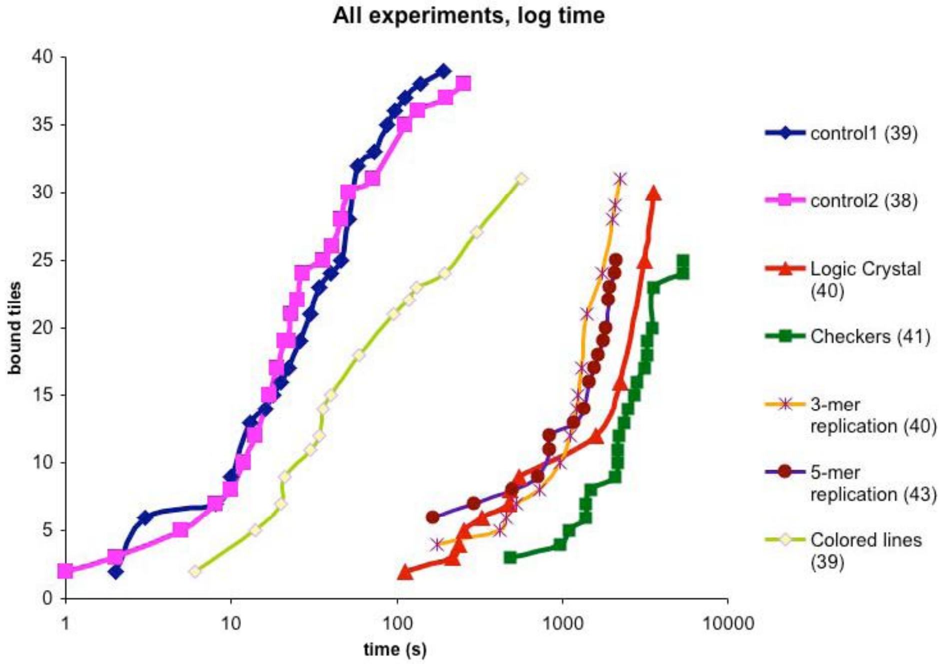


Figure 6-31 All logic limited self assembly experiments graphed together against log time. Number of tiles in reaction vessel shown in (brackets)

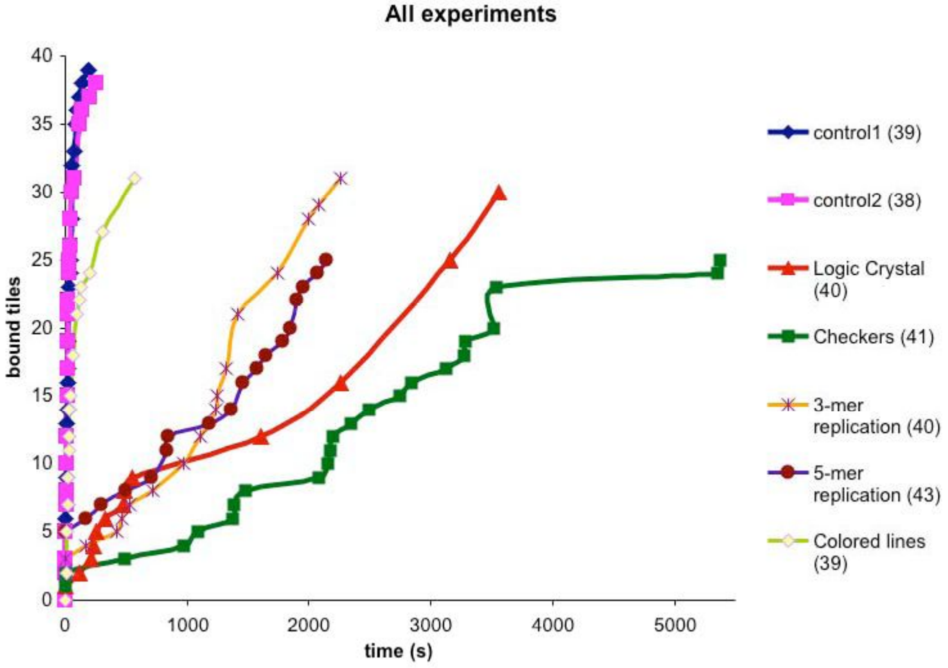


Figure 6-32 All logic limited self assembly experiments graphed together against time.

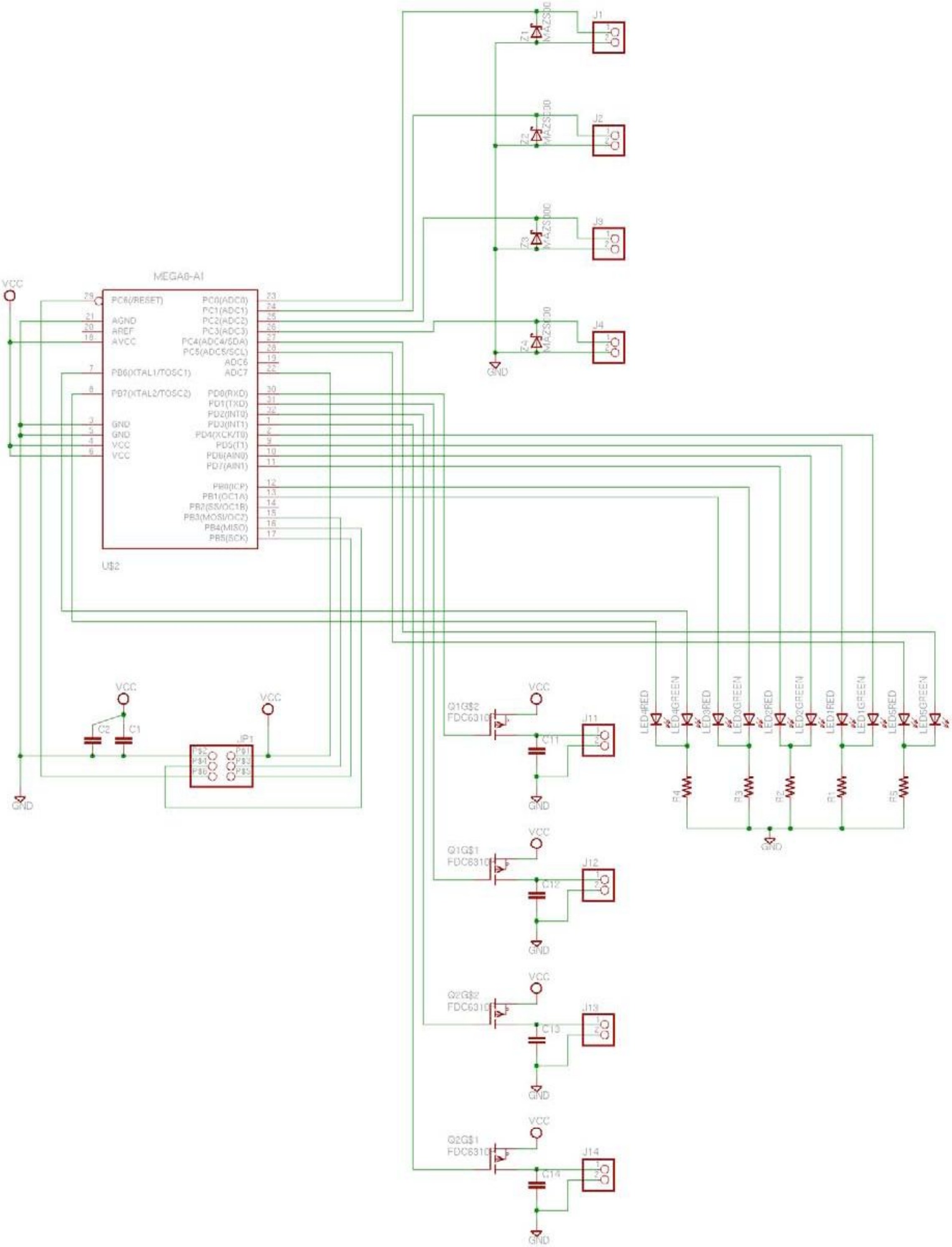


Figure 6-33 Schematic of circuit board used for electromechanical units. (With assistance of Dan Goldwater)