Verifying Concurrent Turing Machines

Soonho Kong Arie Gurfinkel Sagar Chaki

June 4, 2012

Internship Started

working with Arie and Sagar

June 4, 2012

Topic:

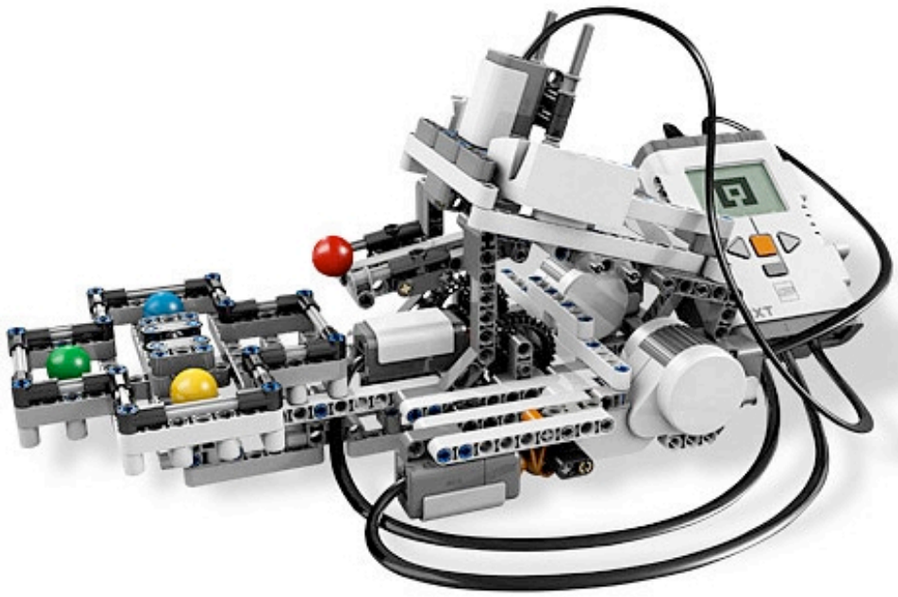
"Time-bounded Analysis of Real-time Systems"

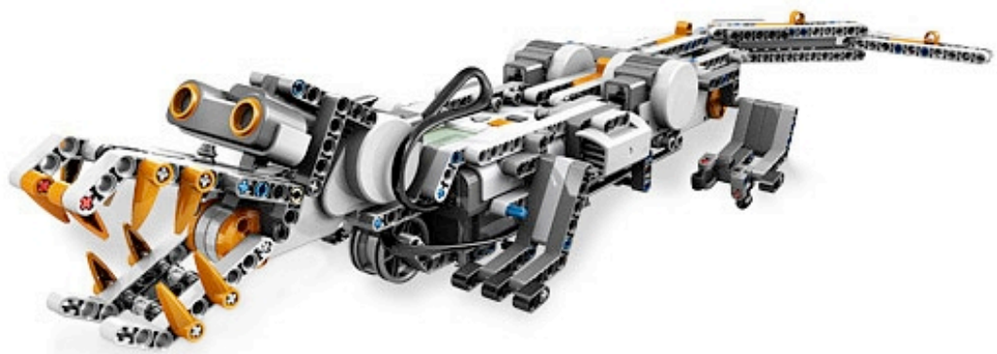
June 4, 2012

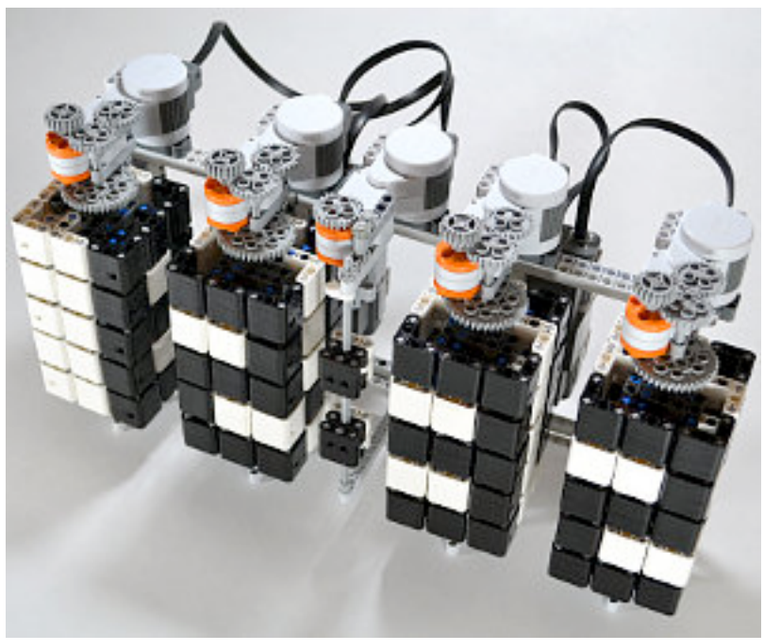
Verification of "Concurrent, Periodic, Real-time Embedded System"

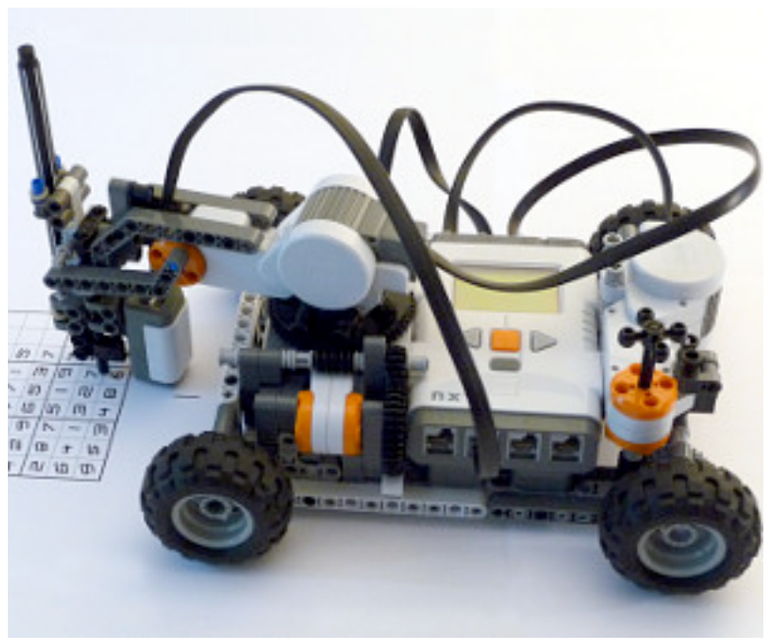
June 5, 2012

We need more examples of concurrent systems.
Can you make one with LEGO MINDSTORMS?







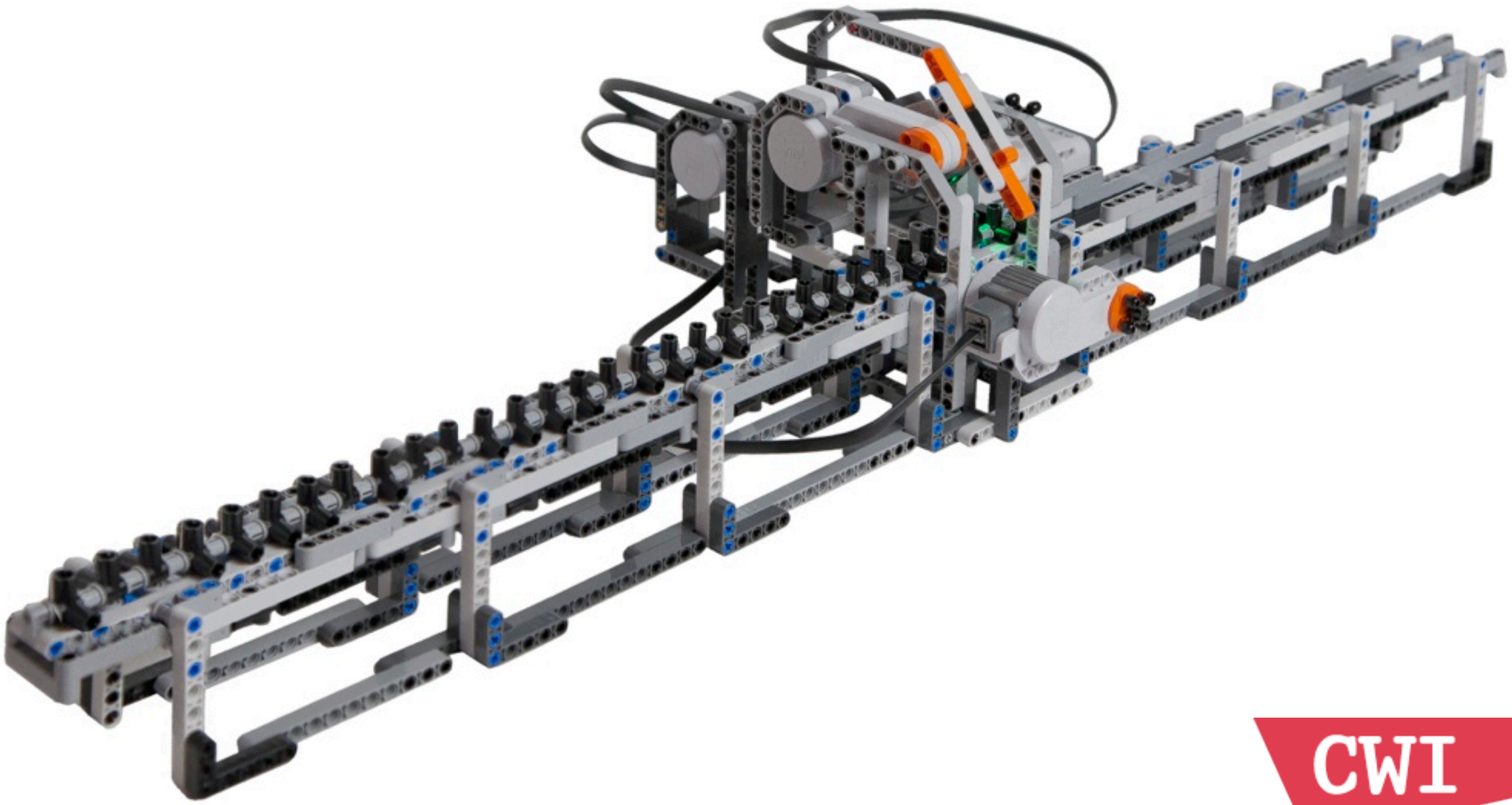


ALAN TURINGYEAR

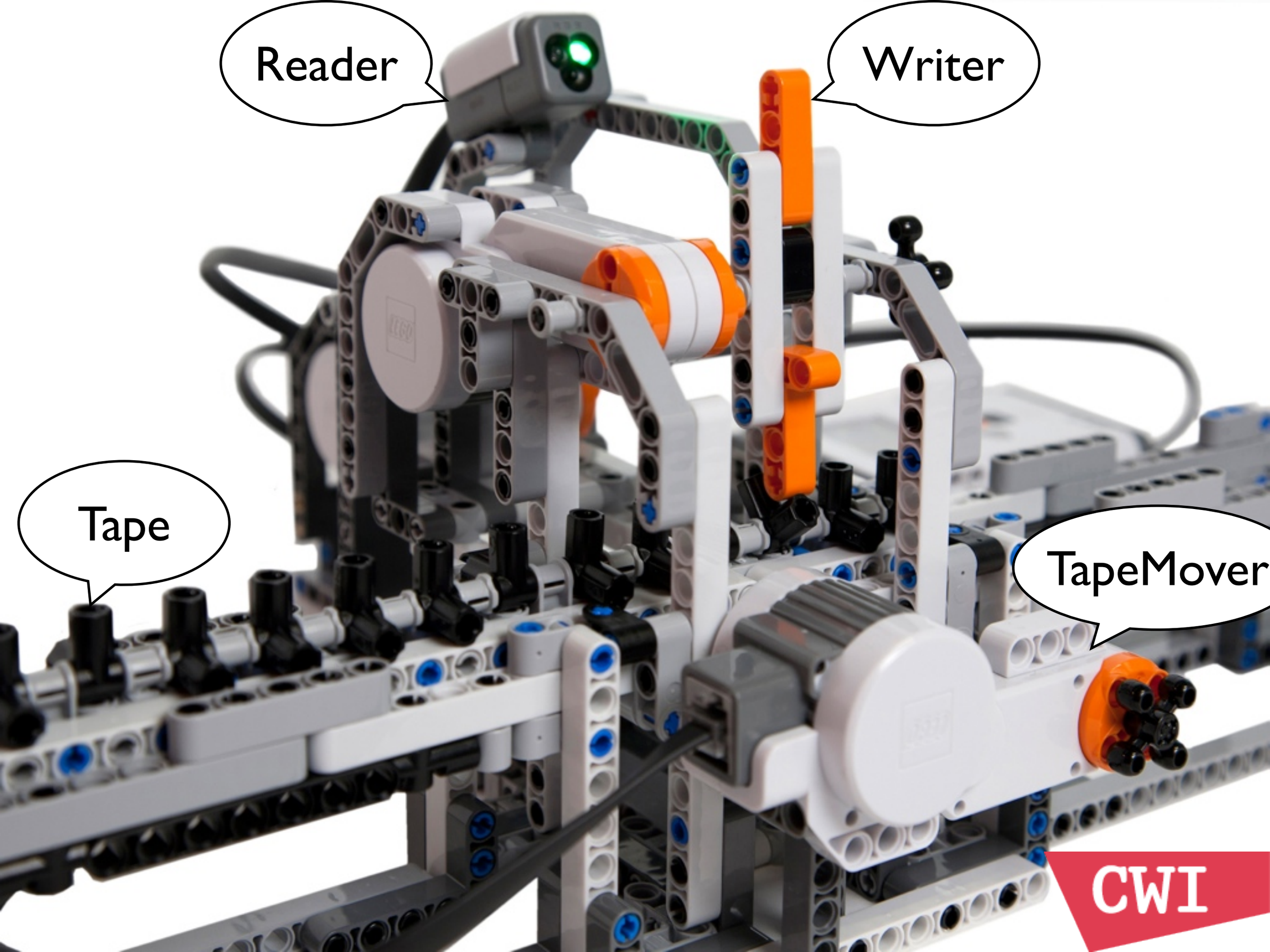
2012

LEGO Turing Machine?

A Turing Machine built using LEGO

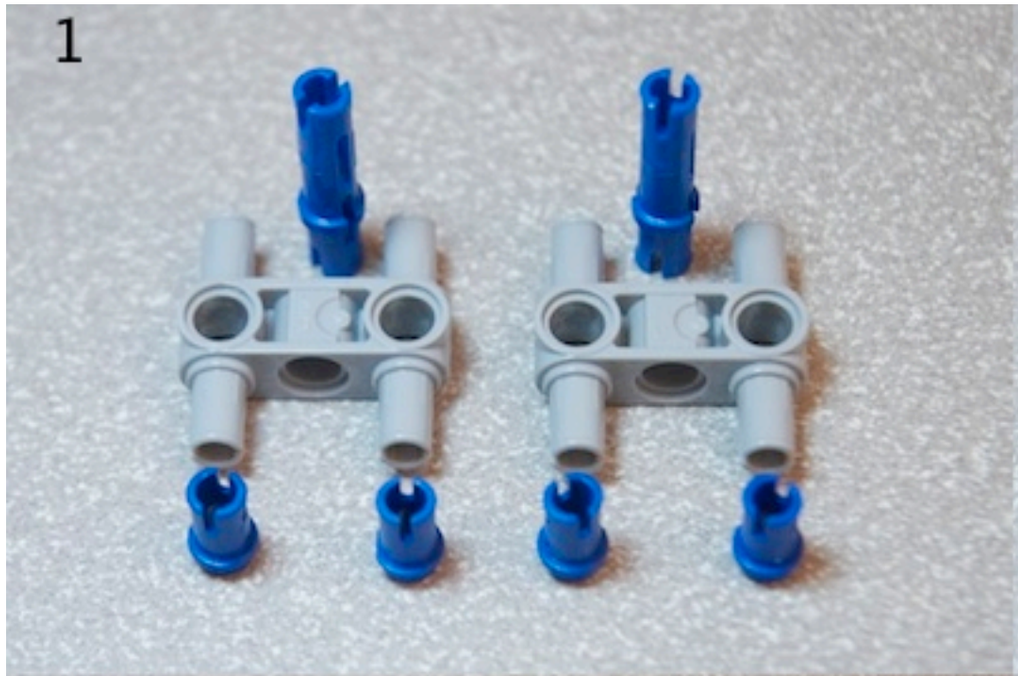


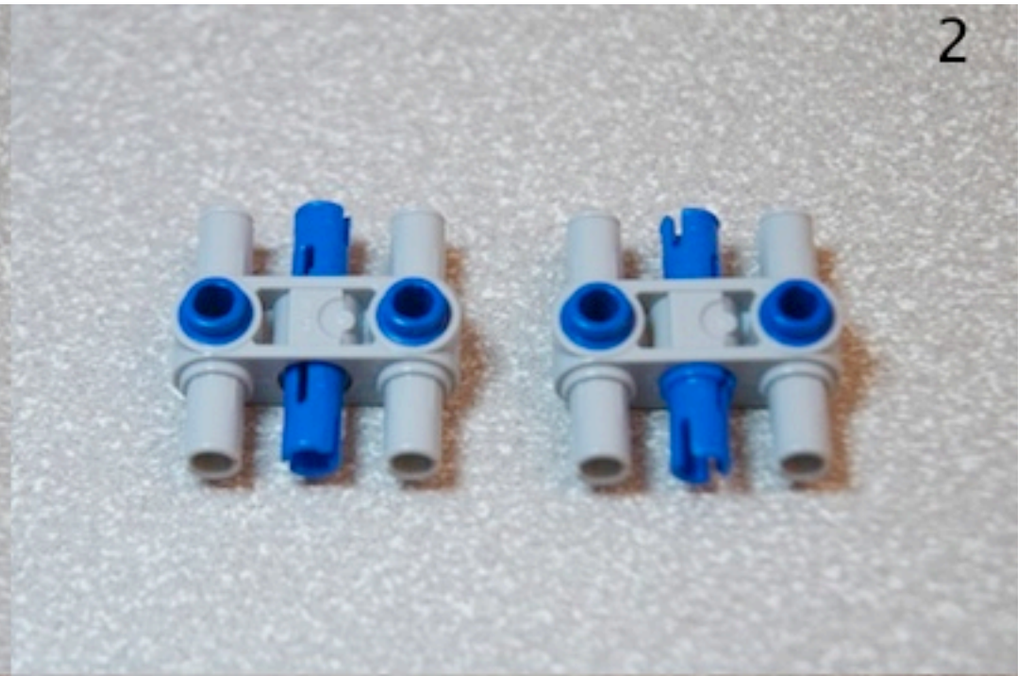
CWI

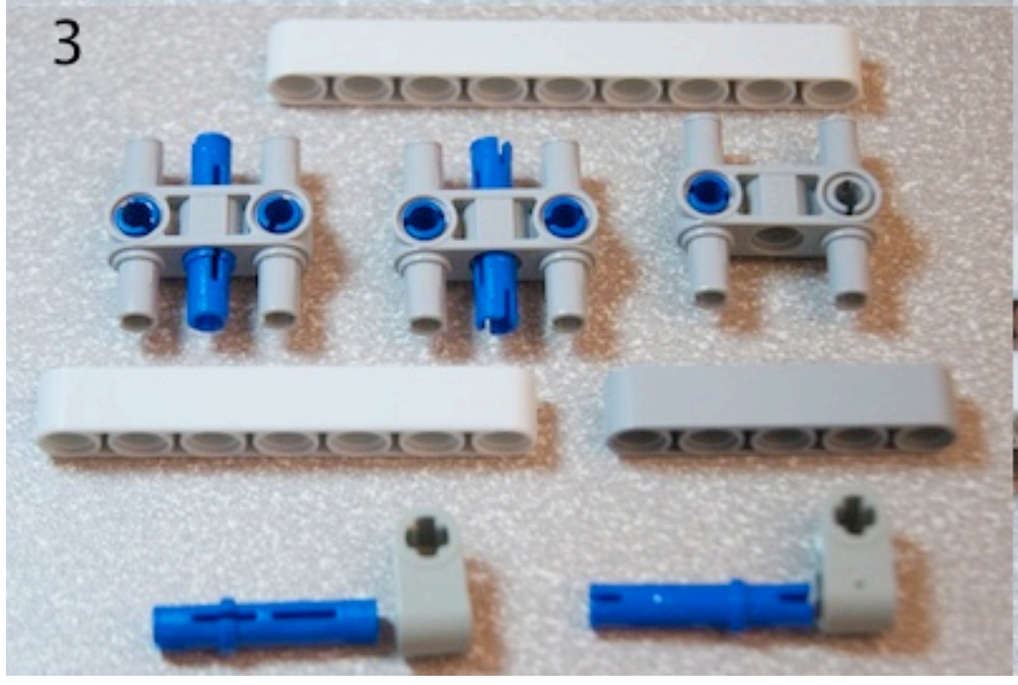

Reader
Writer
Tape
TapeMover
CWI

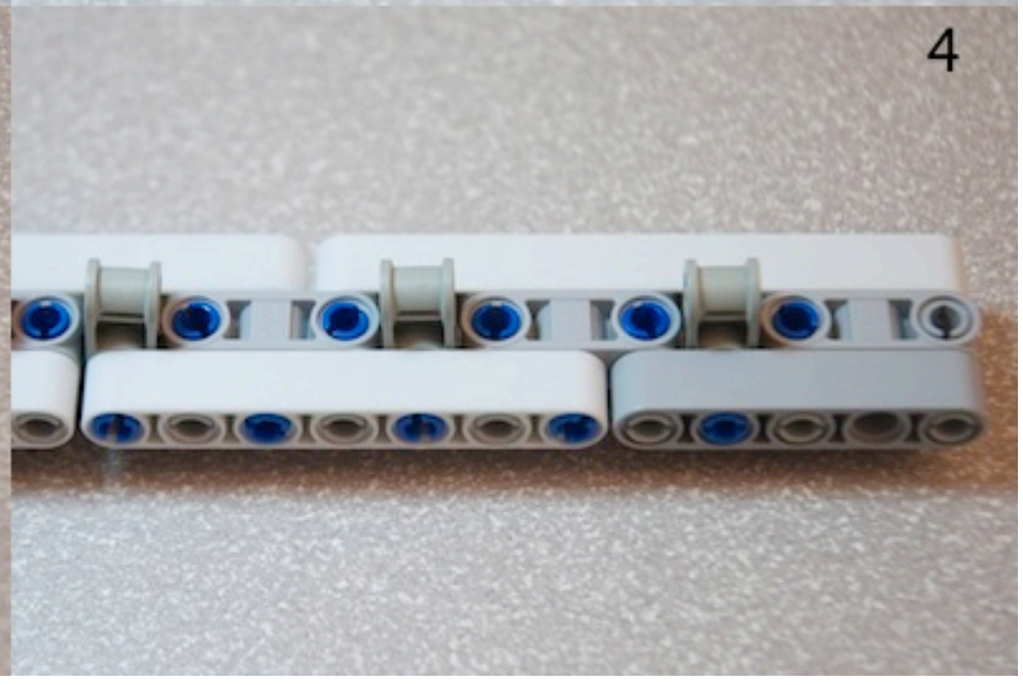
June 9, 2012

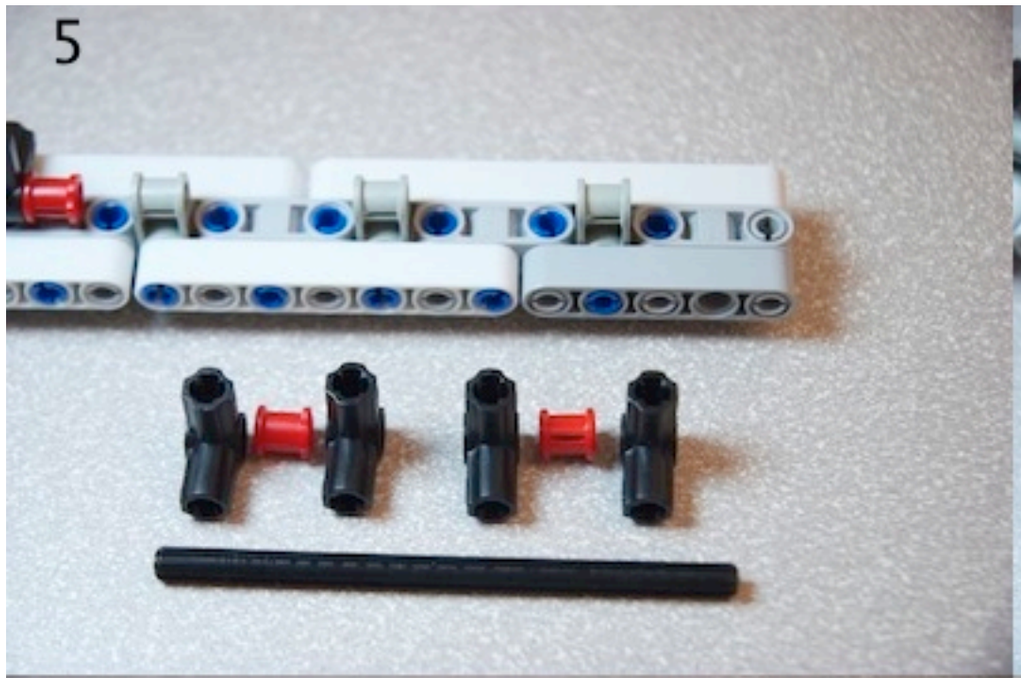
OK, Let's build one!

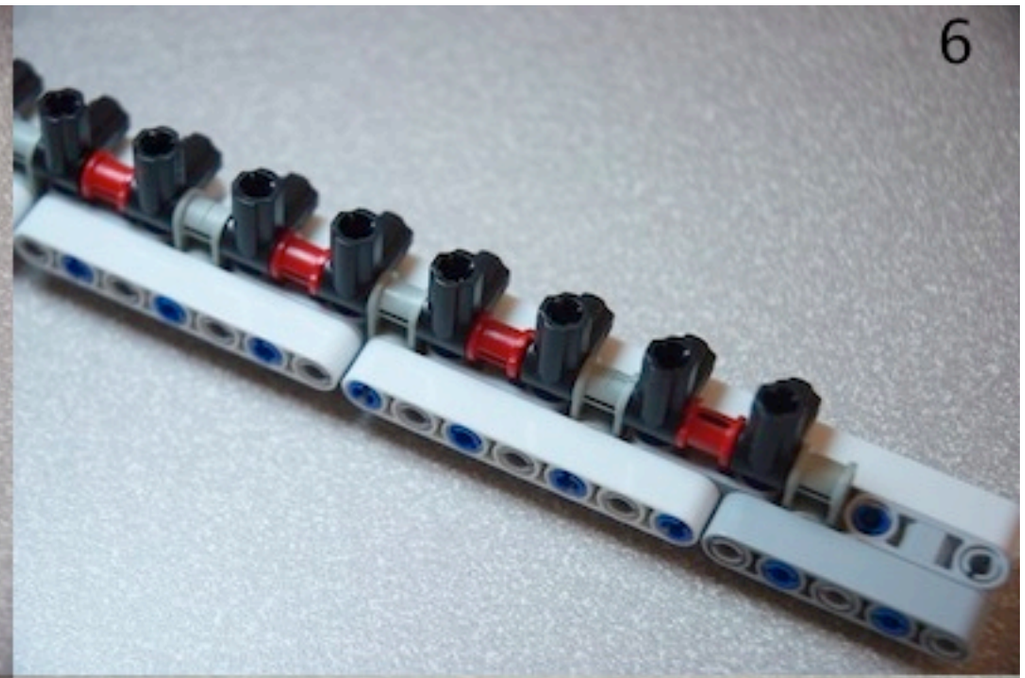

1

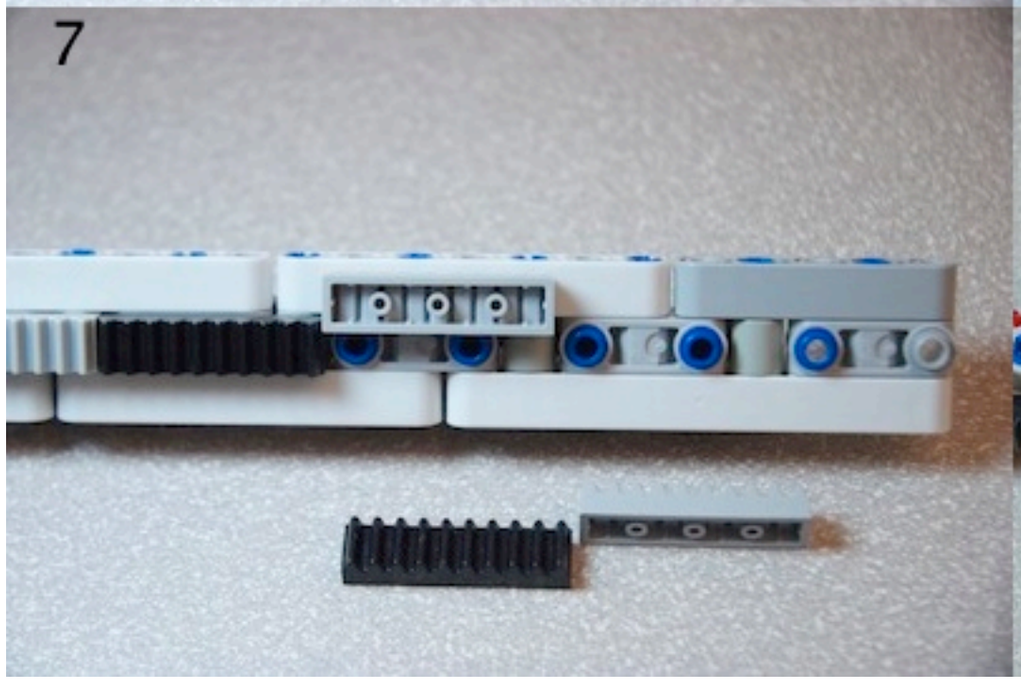

2

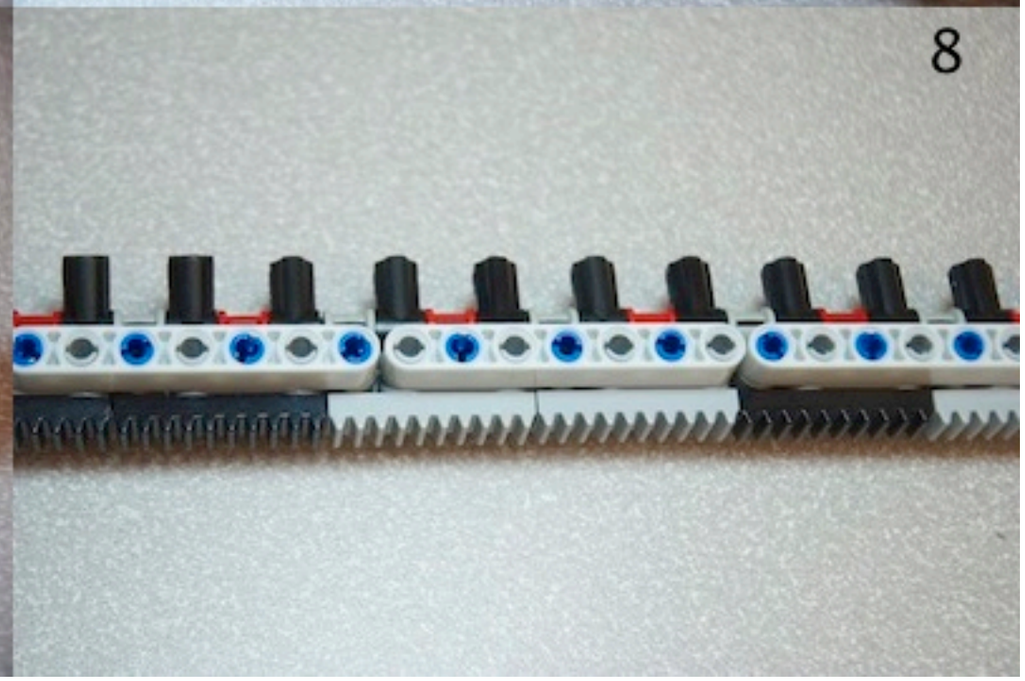

3

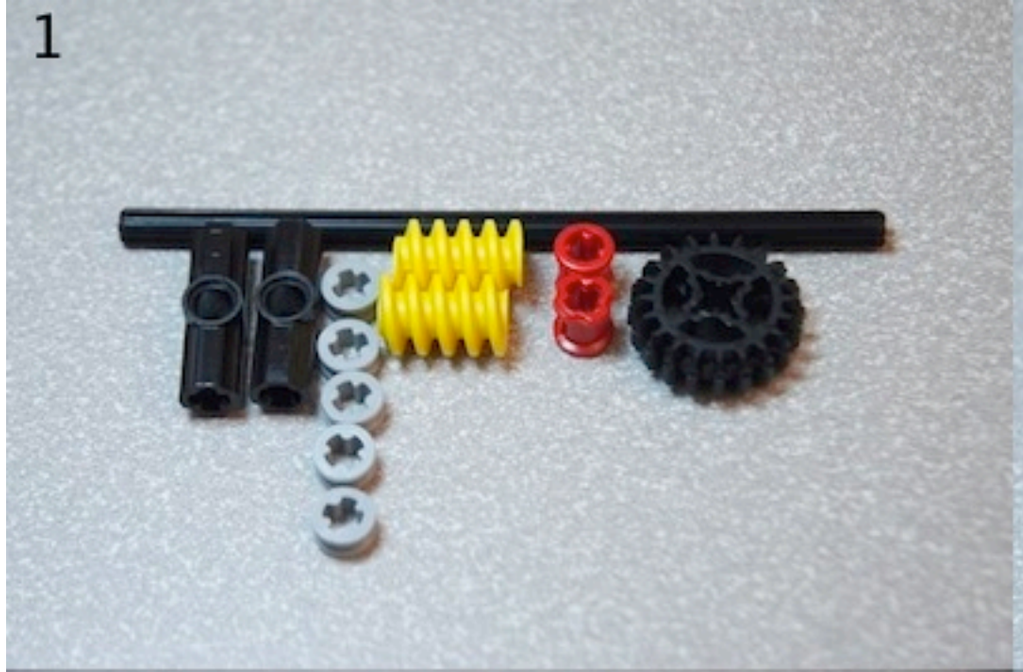

4

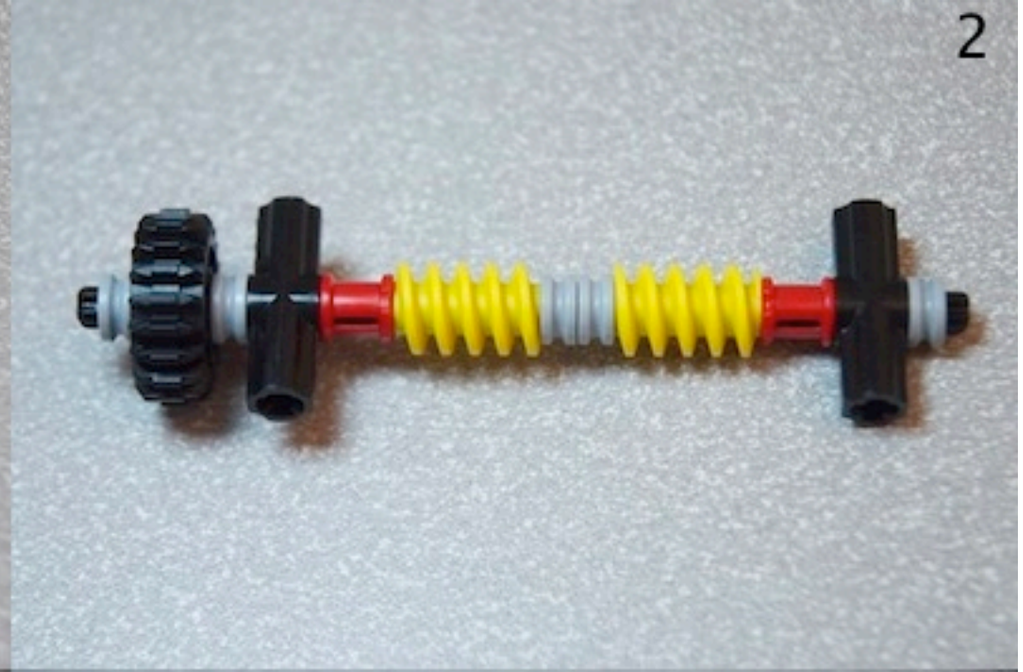

5

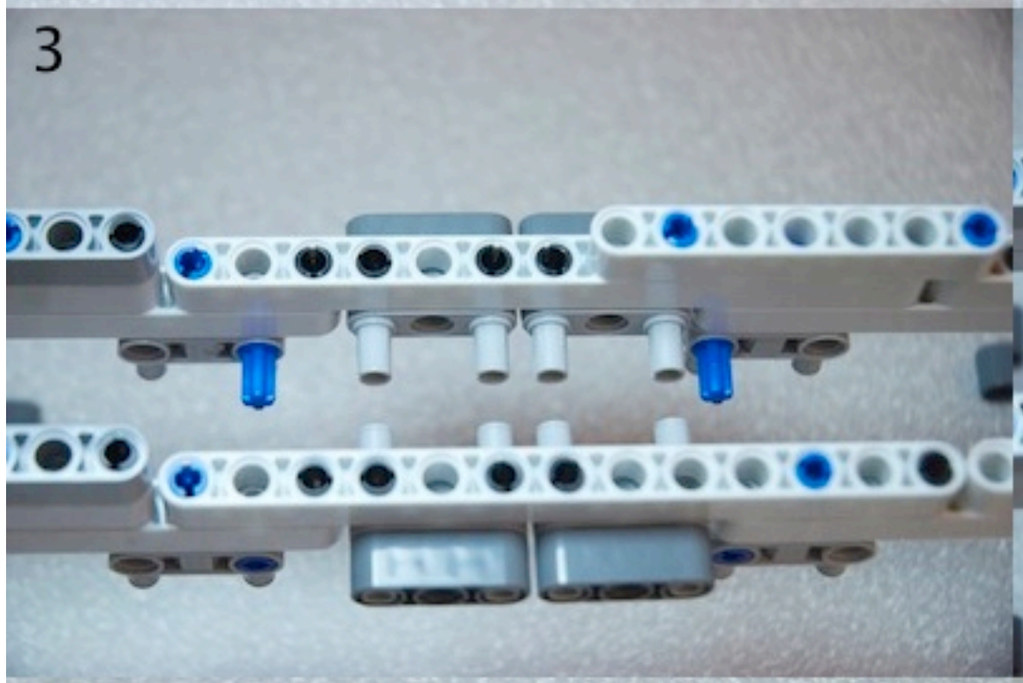

6

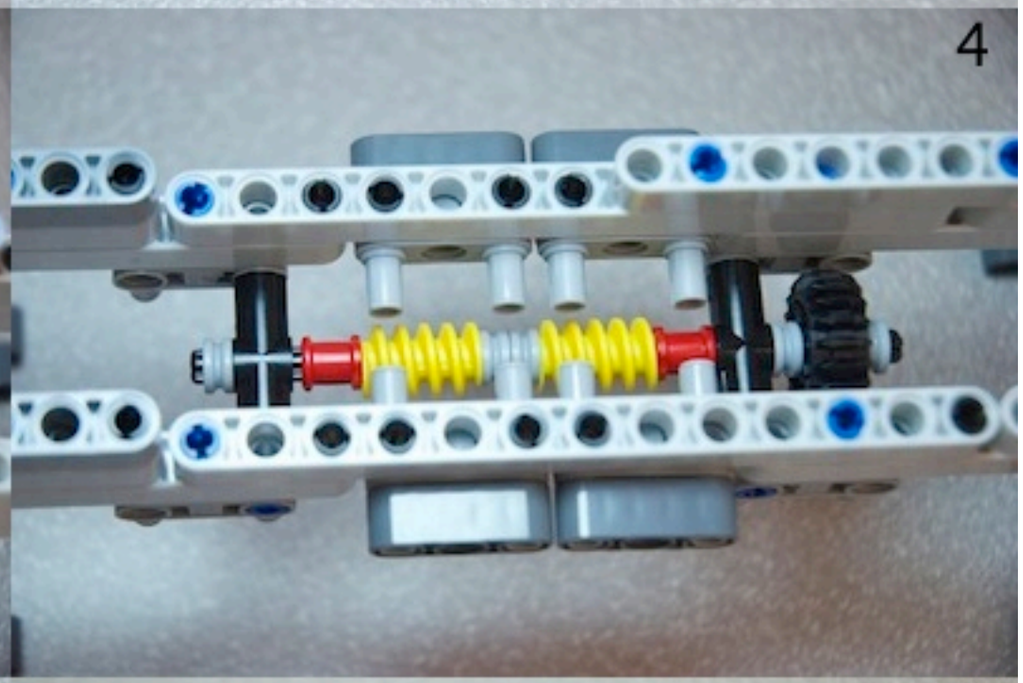

7

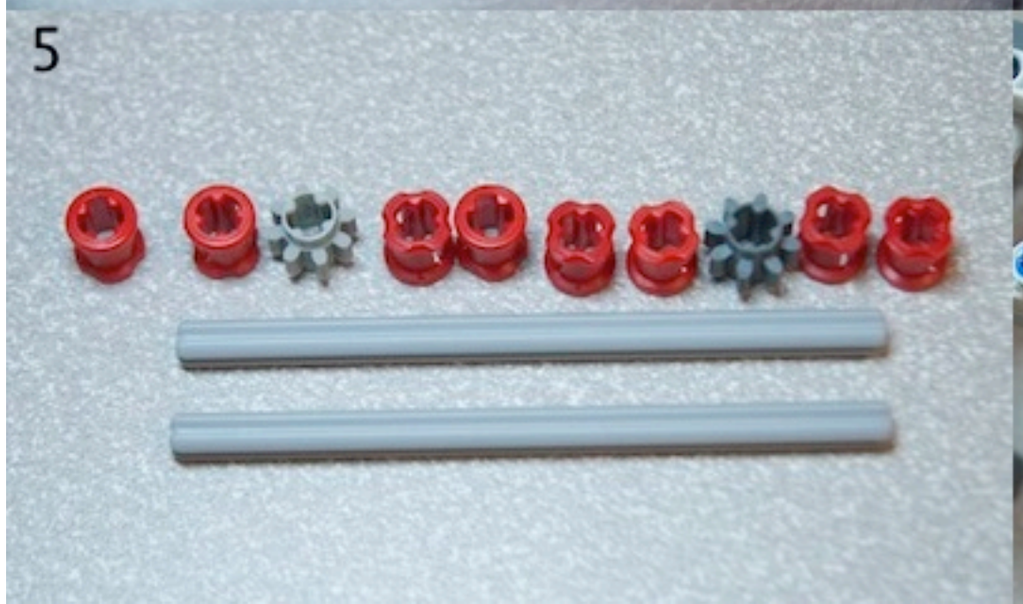

8

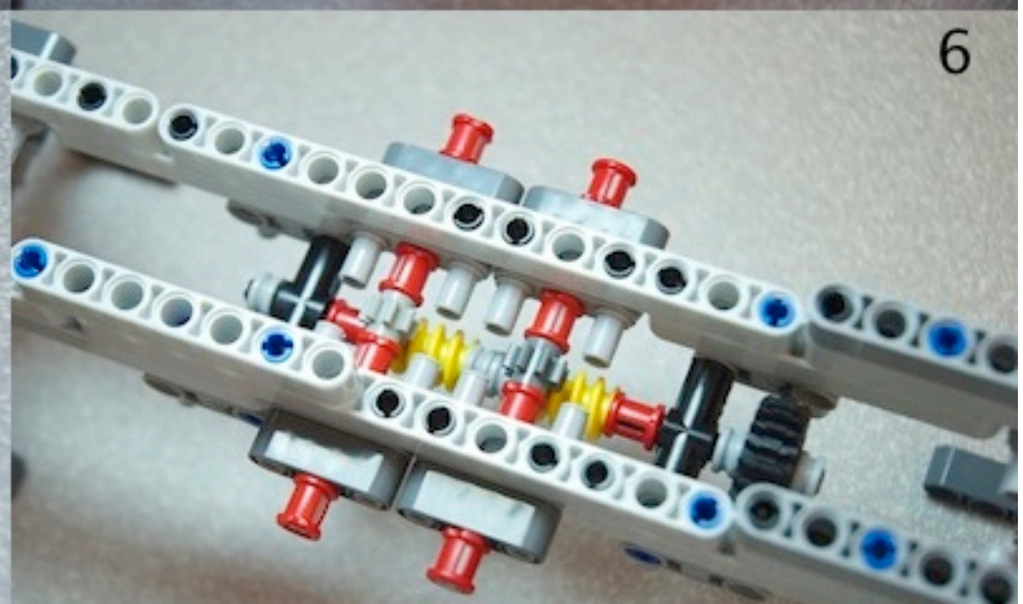

1

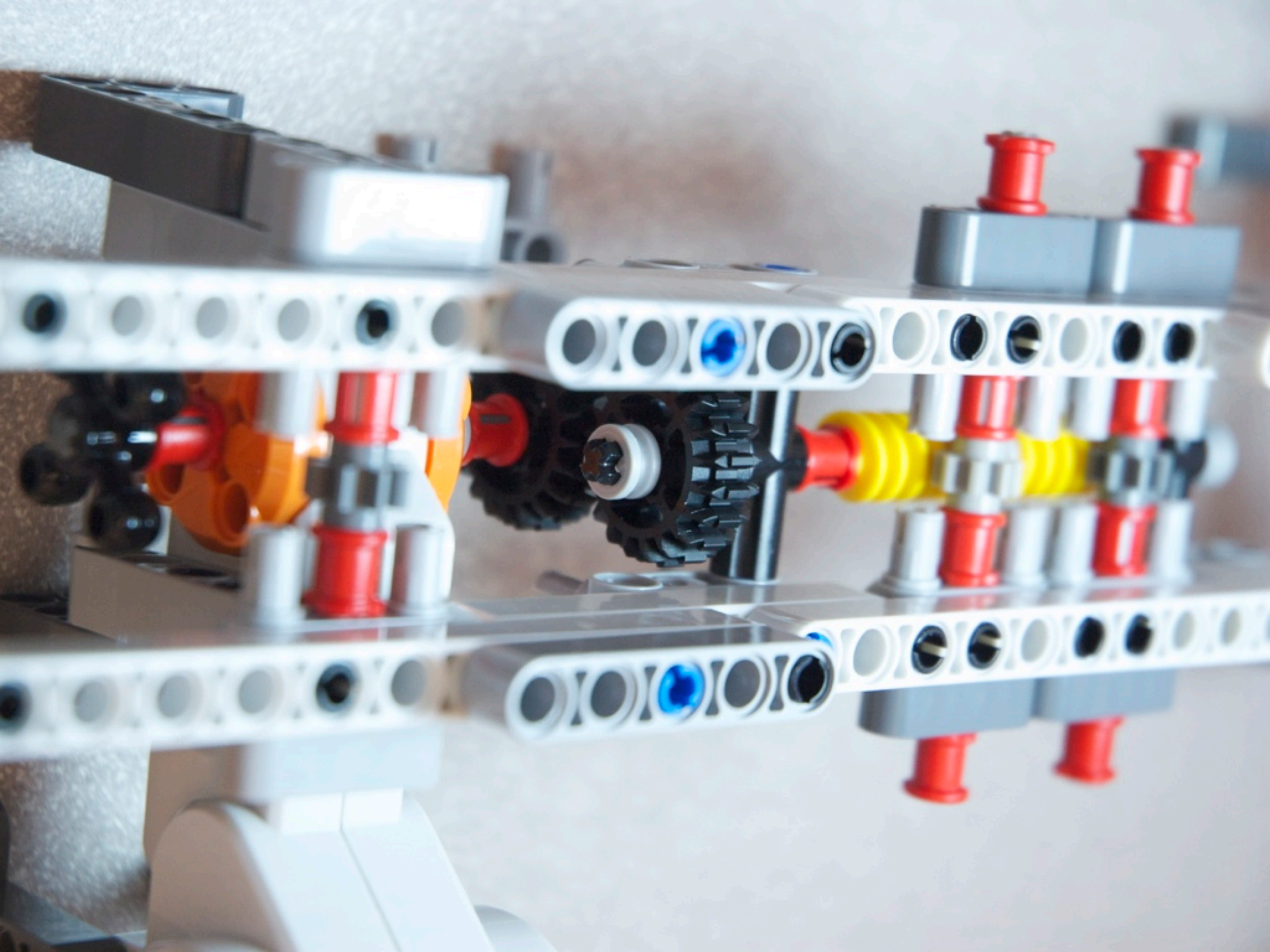

2

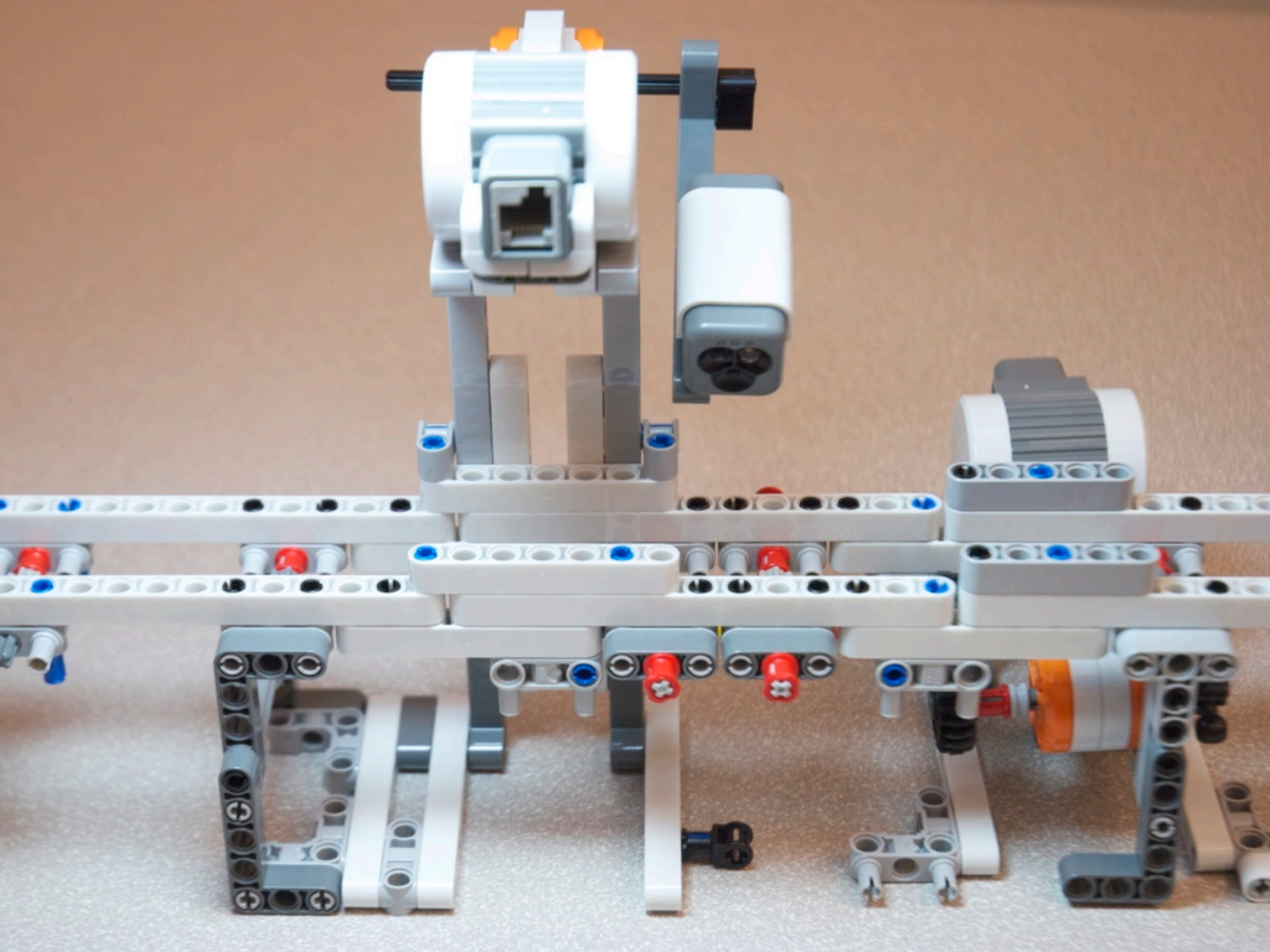

3


4


5


6





July 25, 2012
Construction Completed!

Software Implementation

Writer Task

Priority: 4
Period 25ms | WCET: < 1ms

Flip bits

Controller Task

Priority: 1 (Lowest)
Period 500ms | WCET: 440ms

1. Calibrate Sensor
2. Command other tasks

Reader Task

Priority: 3
Period 50ms | WCET: < 1ms

Read bits using NXT-colorsensor

TapeMover Task

Priority: 2
Period 100ms | WCET: < 1ms

Move the tape (left or right)

DEMO

Unary Addition

2 + 3 = ?

http://www.youtube.com/watch?v=teDyd0d5M4o

Properties

Property 1: When a bit is being read, all the motors should **stop**.

Property 2: When writer flips a bit, the tape motor and read motor should **stop**.

Property 3: When tape moves, the writer motor and read motor should **stop**.

Property 4: When a bit is being read, the sensor should be on **Green** mode

Property 5: The sensor mode must be switched in **Controller Task**, not in Reader Task

Properties

Property 1: When a bit is being read, all the motors should **stop**.

Property 2: When writer flips a bit, the tape motor and read motor should **stop**.

Property 3: When tape moves, the writer motor and read motor should **stop**.

Property 4: When a bit is being read, the sensor should be on **Green** mode

Property 5: Th in Reader Task

```
  case READ_SENSOR:
    if(!R(need_to_run_nxtbg)) {

#ifdef VERIFICATION
      /* Property 1: When a bit is being read,
         all the motors should be stopped.  */
      /* PASSED with 4 hyper-periods */
      assert(R(R_speed) == 0 && R(W_speed) == 0 && R(T_speed) == 0);

      /* Property 4: When a bit is being read,
         the sensor should be on Green mode */
      assert(ecrobot_get_nxtcolorsensor_mode(COLOR_SENSOR) == NXT_LIGHTSENSOR_GREEN);
#endif
      /* Read Sensor Value */
      bg_nxtcolorsensor(false);
      color = ecrobot_get_nxtcolorsensor_light(COLOR_SENSOR);
      W(input, color < R(threshold) ? 1 : 0);
```

Properties

Property 1: When a bit is being read, all the motors should **stop**.

Property 2: When writer flips a bit, the tape motor and read motor should **stop**.

Property 3: When tape moves, the writer motor and read motor should **stop**.

Property 4: When a b mode

Property 5: The sens , not in Reader Task

```
  case WRITE_FLIP:
#ifdef VERIFICATION
    /* Property 3: When writer flips a bit, the tape motor and read
       motor should be stopped. */

    /* IT FAILS!! with BOUND 120 */
    assert(R(T_speed) == 0 && R(R_speed) == 0);
#endif
```

Properties

Property 2: When writer flips a bit, the tape motor and read motor should **stop**.

Controller Task

```
case C_WRITE:

  /* Check if we need to chagne the bit */
  if(R(input) != R(output)) {
    /* Check the header and move it back if necessary */
    if(nxt_motor_get_count(READ_MOTOR) > 0 && R(R_state) == READ_IDLE) {
      W(R_state, READ_MOVE_HEADER_BACKWARD);
    }

    /* Check the header and flip the bit if it is safe to do */
    if(nxt_motor_get_count(READ_MOTOR) <= 0 && R(W_state) == WRITE_IDLE) {
      W(W_state, WRITE_FLIP);
    }
  } else {
    /* Nothing to change for writer */
    W(W_state, WRITE_IDLE);
    C_state = C_MOVE;
  }
  break;
```

Writer Task

```
  case WRITE_FLIP:
#ifdef VERIFICATION
    /* Property 3: When writer flips a bit, the tape motor and read
       motor should be stopped. */

    /* IT FAILS!! with BOUND 120 */
    assert(R(T_speed) == 0 && R(R_speed) == 0);
#endif
```

Properties

Property 2: When w motor and read motor should **stop**.

Do we need to write?

Controller Task

```
case C_WRITE:

  /* Check if we need to chagne the bit */
  if(R(input) != R(output)) {
    /* Check the header and move it back if necessary */
    if(nxt_motor_get_count(READ_MOTOR) > 0 && R(R_state) == READ_IDLE) {
      W(R_state, READ_MOVE_HEADER_BACKWARD);
    }

    /* Check the header and flip the bit if it is safe to do */
    if(nxt_motor_get_count(READ_MOTOR) <= 0 && R(W_state) == WRITE_IDLE) {
      W(W_state, WRITE_FLIP);
    }
  } else {
    /* Nothing to change for writer */
    W(W_state, WRITE_IDLE);
    C_state = C_MOVE;
  }
  break;
```

Writer Task

```
  case WRITE_FLIP:
#ifdef VERIFICATION
    /* Property 3: When writer flips a bit, the tape motor and read
       motor should be stopped. */

    /* IT FAILS!! with BOUND 120 */
    assert(R(T_speed) == 0 && R(R_speed) == 0);
#endif
```

Properties

Property 2: When writer flips a bit, the tape motor and read motor should **stop**.

Controller Task

```
case C_WRITE:

  /* Check if we need to chagne the bit */
  if(R(input) != R(output)) {
    /* Check the header and move it back if necessary */
    if(nxt_motor_get_count(READ_MOTOR) > 0 && R(R_state) == READ_IDLE) {
      W(R_state, READ_MOVE_HEADER_BACKWARD);
    }

    /* Check the header and flip the bit if it is safe to do */
    if(nxt_motor_get_count(READ_MOTOR) <= 0 && R(W_state) == WRITE_IDLE) {
      W(W_state, WRITE_FLIP);
    }
  } else {
    /* Nothing to change for writer */
    W(W_state, WRITE_IDLE);
    C_state = C_MOVE;
  }
  break;
```

If the READ header is up,
Move it back
to avoid collision!

Writer Task

```
  case WRITE_FLIP:
#ifdef VERIFICATION
    /* Property 3: When writer flips a bit, the tape motor and read
       motor should be stopped. */

    /* IT FAILS!! with BOUND 120 */
    assert(R(T_speed) == 0 && R(R_speed) == 0);
#endif
```

Properties

Property 2: When writer flips a bit, the tape motor and read motor should **stop**.

Controller Task

```
case C_WRITE:

  /* Check if we need to chagne the bit */
  if(R(input) != R(output)) {
    /* Check the header and move it back if necessary */
    if(nxt_motor_get_count(READ_MOTOR) > 0 && R(R_state) == READ_IDLE) {
      W(R_state, READ_MOVE_HEADER_BACKWARD);
    }

    /* Check the header and flip the bit if it is safe to do */
    if(nxt_motor_get_count(READ_MOTOR) <= 0 && R(W_state) == WRITE_IDLE) {
      W(W_state, WRITE_FLIP);
    }
  } else {
    /* Nothing to change for writer */
    W(W_state, WRITE_IDLE);
    C_state = C_MOVE;
  }
  break;
```

OK, it's safe to write!

Writer Task

```
  case WRITE_FLIP:
#ifdef VERIFICATION
    /* Property 3: When writer flips a bit, the tape motor and read
       motor should be stopped. */

    /* IT FAILS!! with BOUND 120 */
    assert(R(T_speed) == 0 && R(R_speed) == 0);
#endif
```

Properties

Property 2: When writer flips a bit, the tape motor and read motor should **stop**.

Controller Task

```
case C_WRITE:

  /* Check if we need to chagne the bit */
  if(R(input) != R(output)) {
    /* Check the header and move it back if necessary */
    if(nxt_motor_get_count(READ_MOTOR) > 0 && R(R_state) == READ_IDLE) {
      W(R_state, READ_MOVE_HEADER_BACKWARD);
    }

    /* Check the header and flip the bit if it is safe to do */
    if(nxt_motor_get_count(READ_MOTOR) <= 0 && R(W_state) == WRITE_IDLE) {
      W(W_state, WRITE_FLIP);
    }
  } else {
    /* Nothing to change for writer */
    W(W_state, WRITE_IDLE);
    C_state = C_MOVE;
  }
  break;
```

Writer Task

```
  case WRITE_FLIP:
#ifdef VERIFICATION
    /* Property 3: When writer flips a bit, the tape motor and read
       motor should be stopped. */

    /* IT FAILS!! with BOUND 120 */
    assert(R(T_speed) == 0 && R(R_speed) == 0);
#endif
```

NO!
The position of READ header is in safe area (<=0), however it's possible that it is **still moving**!

Properties

Property 2: When writer flips a bit, the tape motor and read motor should **stop**.

Controller Task

```
case C_WRITE:

  /* Check if we need to chagne the bit */
  if(R(input) != R(output)) {
    /* Check the header and move it back if necessary */
    if(nxt_motor_get_count(READ_MOTOR) > 0 && R(R_state) == READ_IDLE) {
      W(R_state, READ_MOVE_HEADER_BACKWARD);
    }

    /* Check the header and flip the bit if it is safe to do */
    if(nxt_motor_get_count(READ_MOTOR) <= 0 && R(W_state) == WRITE_IDLE) {
      W(W_state, WRITE_FLIP);
    }
  } else {
    /* Nothing to change for writer */
    W(W_state, WRITE_IDLE);
    C_state = C_MOVE;
  }
  break;
```

Writer Task

```
  case WRITE_FLIP:
#ifdef VERIFICATION
    /* Property 3: When writer flips a bit, the tape motor and read
       motor should be stopped. */

    /* IT FAILS!! with BOUND 120 */
    assert(R(T_speed) == 0 && R(R_speed) == 0);
#endif
```

REKH(out tool) can find this bug within 2mins.

DEMO

REKH & Counterexample

http://www.cs.cmu.edu/~soonhok/rekh-viz

Thank you!