# PWM Speed Ramping

ELN9207 (Final Project)

April 22th, 2014

Justin Hempell

David Manouchehri

# TABLE OF CONTENTS

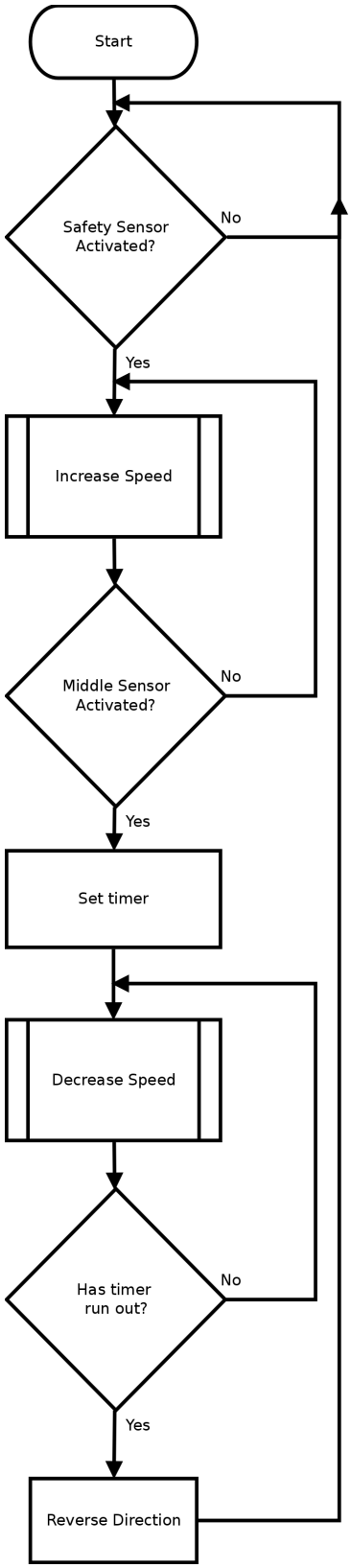
# DESCRIPTION

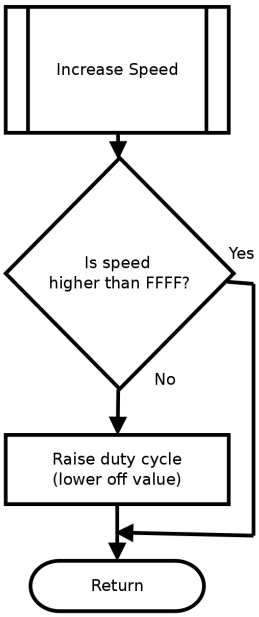
Our program is designed to move an object across a conveyor belt, and vary the speed using pulse width modulation. The object will slow down when nearing the end of the conveyor belt to prevent the object from tipping over. The same concept is applied when starting; it will begin at a slow speed, and slowly accelerate. The conveyor will continue to increase speed until it reaches its maximum speed in the middle (triggered by an optical sensor), at which point it will begin to decelerate. At each end of the conveyor, the program is set up to automatically switch from forward to reverse; a waiting period could be added on either side to allow another process (such as painting) to take place.

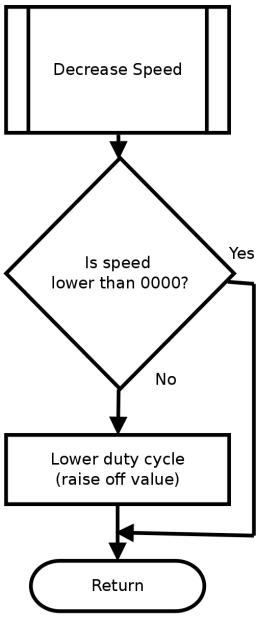
To use the program, an object must be placed in front of the safety proximity sensor. After five seconds, the program will begin to run. This delay is to allow the operator to safely distance themselves from the machine and allow the object to settle.

Another feature of the program is a speedometer, which is displayed using LEDs. The speed is generated by a shift register.

# FLOWCHART


Start
Safety Sensor Activated?
No
Yes
Increase Speed
Middle Sensor Activated?
No
Yes
Set timer
Decrease Speed
Has timer run out?
No
Yes
Reverse Direction


Increase Speed
Is speed higher than FFFF?
Yes
No
Raise duty cycle (lower off value)
Return


Decrease Speed
Is speed lower than 0000?
Yes
No
Lower duty cycle (raise off value)
Return

# LADDER DIAGRAM

The full ladder diagram can be found in the .cxp file which should be attached to this report.

# NEW INSTRUCTIONS: ++B AND --B

In our program we used both the BCD increment and decrement instruction blocks; while we used the ++B in our sequencers, we did not use --B. *(Note: Originally at the time of programming this project, we had learned neither function blocks in our coursework.)* The function of these blocks is to increment or decrement a word every cycle that the block is active for. We used these blocks in our program to adjust the off time of the clock pulse used for PWM. This allowed an easy and convenient way of adjusting the speed at will. We opted to use the standard block instead of the differential up (@) blocks, as our subroutine provides other functionality that prevents the function from continuously running.

Below is the datasheet for ++B; --B is not included as the usage is the same (except decrementing instead of incrementing).

+ +B/+ +BL

| Instruction | Mnemonic | Variations | Function code | Function |
|---|---|---|---|---|
| INCREMENT BCD | + +B | @+ +B | 594 | Increments the 4-digit BCD content of the specified word by 1. |
| DOUBLE INCREMENT BCD | + +BL | @+ +BL | 595 | Increments the 8-digit BCD content of the specified words by 1. |

| | | |
|---|---|---|
| Symbol | + +B   | + +BL   |

Applicable Program Areas

| Area | Function block definitions | Block program areas | Step program areas | Subroutines | Interrupt tasks | SFC action or transition programs |
|---|---|---|---|---|---|---|
| Usage | OK | OK | OK | OK | OK | OK |

Operands

| Operand | Description | Data type | | Size | |
|---|---|---|---|---|---|
| | | ++ | ++L | ++ | ++L |

| | | | | |
|---|---|---|---|---|
| Wd | ++B: Word<br>++BL: First word | WORD | DWORD | 1 | 2 |

I Operand Specifications

| Area | Word addresses | | | | | | | | Indirect DM/EM addresses | | Constants | Registers | | | Flags | | Pulse bits | TR bits |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CIO | WR | HR | AR | T | C | DM | EM | @DM @EM | *DM *EM | | DR | IR | Indirect using IR | TK | CF | | |
| + +B Wd | OK | OK | OK | OK | OK | OK | OK | OK | OK | OK | --- | OK | --- | OK | --- | --- | --- | --- |
| + +BL Wd | | | | | | | | | | | | --- | | | | | | |

Flags

| Name | Label | Operation |
|---|---|---|
| Error Flag | ER | • ON if the content of Wd/Wd+1 and Wd is not BCD.<br>• OFF in all other cases. |
| Equals Flag | = | • ON if the result is 0000/0000 0000 after execution.<br>• OFF in all other cases. |
| Carry Flag | CY | • ON if a digit in Wd/Wd+1 or Wd went from 9 to 0 during execution.<br>• OFF in all other cases. |

Function

I + +B

The + +B(594) instruction adds 1 to the BCD content of Wd. The specified word will be incremented by 1 every cycle as long as the execution condition of + +B(594) is ON. When the up-differentiated variation of this instruction (@+ +B(594)) is used, the specified word is incremented only when the execution condition has gone from OFF to ON.


I + +BL

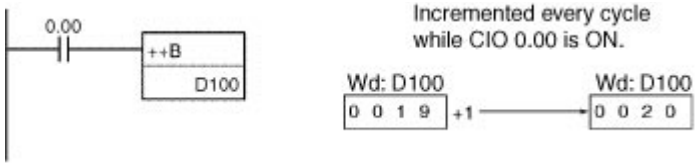
The + +BL(595) instruction adds 1 to the 8-digit BCD content of Wd+1 and Wd. The content of the specified words will be incremented by 1

every cycle as long as the execution condition of + +BL(595) is ON. When the up-differentiated variation of this instruction (@+ +BL(595)) is used, the content of the specified words is incremented only when the execution condition has gone from OFF to ON.

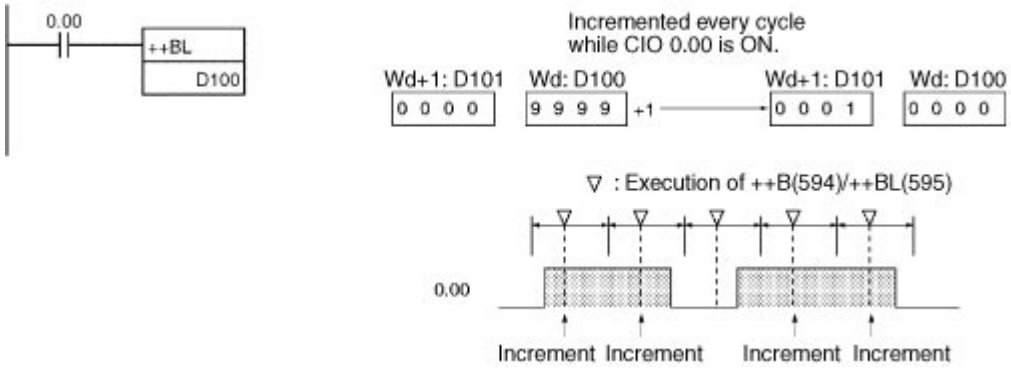
Sample program

l Operation of + +B(594)/+ +BL(595)

In the following example, the BCD content of D100 will be incremented by 1 every cycle as long as CIO 0.00 is ON.



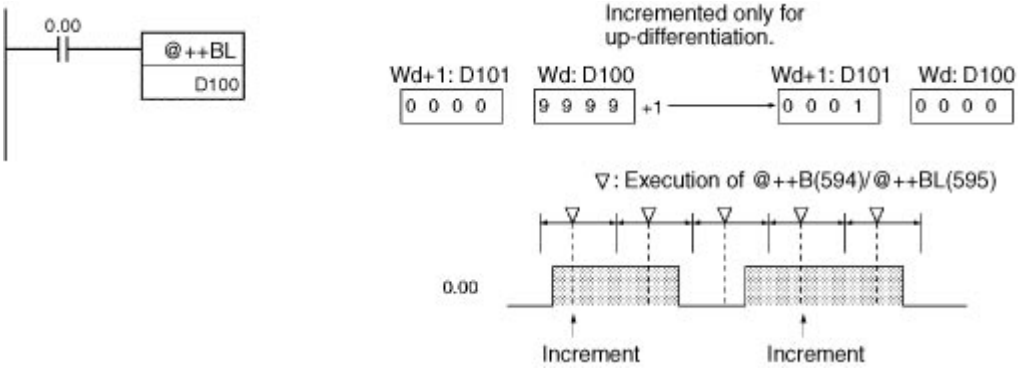

In the following example, the 8-digit BCD content of D101 and D100 will be incremented by 1 every cycle as long as CIO 0.00 is ON.




l Operation of @+ +B(594)/@+ +BL(595)

The up-differentiated variation is used in the following example, so the content of D100 will be incremented by 1 only when CIO 0.00 has gone from OFF to ON.


The up-differentiated variation is used in the following example, so the BCD content of D101 and D100 will be incremented by 1 only when CIO 0.00 has gone from OFF to ON.




Availability:

[Supported PLCs]

## SUBROUTINE

The subroutines in our program are responsible for increasing and decreasing the off time in the clock pulse using the incrementing and decrementing BCD instruction blocks (which were explained in the preivous section). When the object on the conveyor passes the middle sensor, the incrementing subroutine is called to increase the value at address W1; this will increase off time and slow down the conveyor. Using a compare data instruction block, the value of W1 must be less than FFFF for the value to increment. Without this, the value would rollover. A 0.2 second clock pulse input is on the same rung, which will increment the value at W1 by 1 every 0.2 seconds on the differential up (this is why @++B is not needed). Whenever there is a change in direction of the conveyor belt W0.12 will be reset to 0 activating the decrementing subroutine. The decrementing subroutine works exactly the same as the incrementing, only it is decreasing off time at W1 and increasing the motor speed. The value at W0.12 will decide which subroutine is activated. When the middle sensor is activated the value is held by a keep instruction. The NO contact is closed which will activate the incrementing subroutine. The NC contact is now open which will deactivate the decrementing subroutine. The keep instruction will be reset to 0 any time the conveyor changes direction. This is done with differentiate up inputs for forward and reverse connected to the reset input of the keep instruction. When the conveyor changes directions the value at W0.12 will be reset to 0 which activated the decrementing subroutine, increasing the motor speed. Again a 0.2 second clock pulse is used on the rung along with a compare data instruction block. The value at W1 must be greater than 0000 for the value to decrement; it could be set to a higher number if the operator wanted to avoid running at full speed at all times.

## SYMBOLS TABLE AND DATA MEMORY AREA

Can be found in the attached .cxp file. Data memory area was not modified.

## LIST OF EQUIPMENT

- 1x CJ1M/CJ2M (cross compatible)
- 1x Conveyor Belt
- 2x External Relays
- 1x Optical Sensor
- 1x Proximity Sensor

## DRAWING OF PROCESS

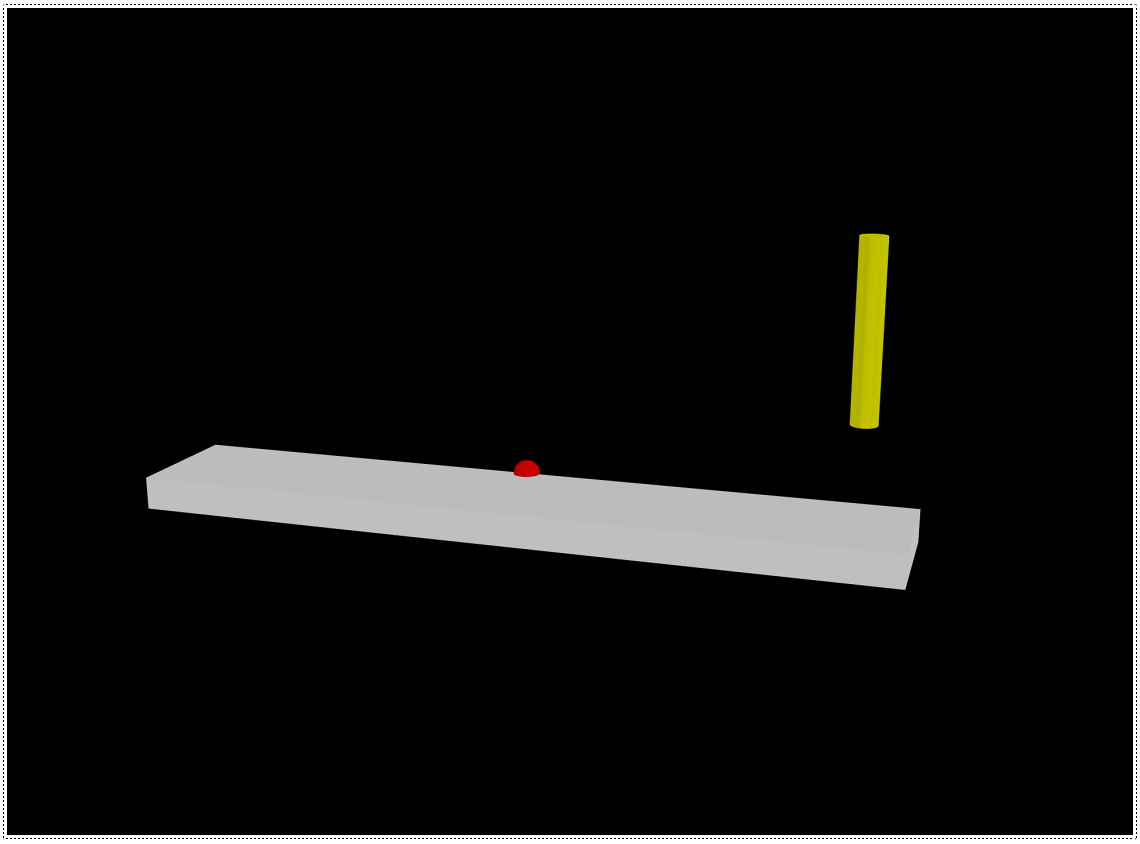


*Illustration 1: The gray box represents the conveyor, the red circle represents the optical sensor, and the yellow cylinder represents the proximity sensor.*

# DEMONSTRATION

*Note: The safety start sensor and status LEDs were not implemented in this example.*

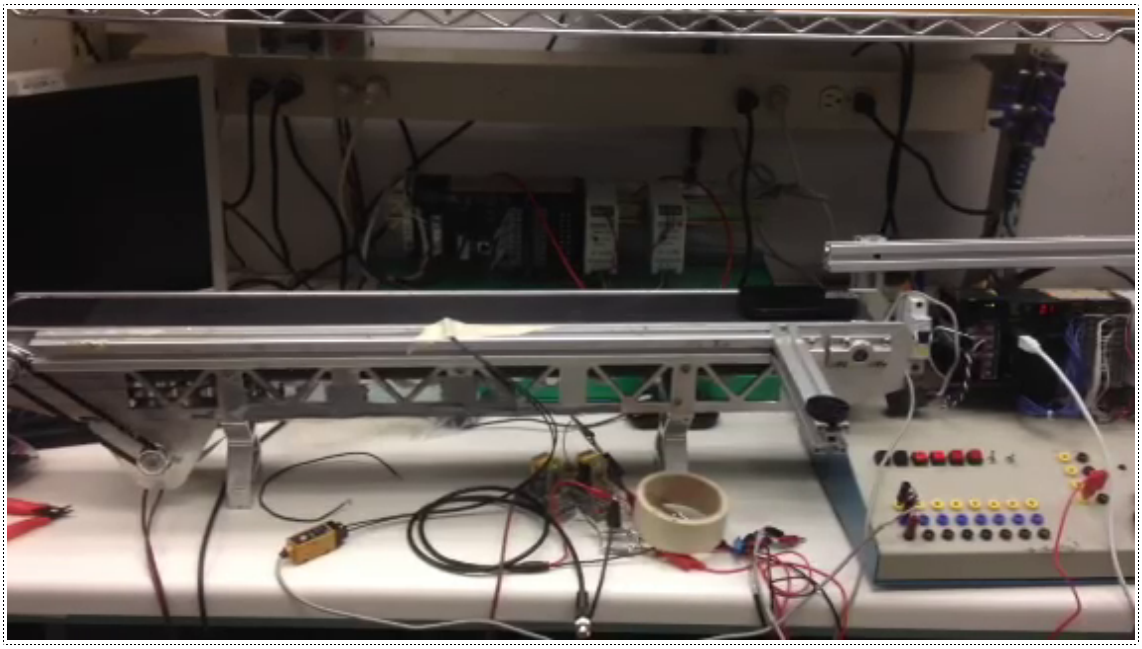


*Illustration 2: An object is placed on the far right of the conveyor belt. It will begin to move to the left (forward) if the enable switch is toggled. It will be begin at a slow pace, and slowly speed up.*

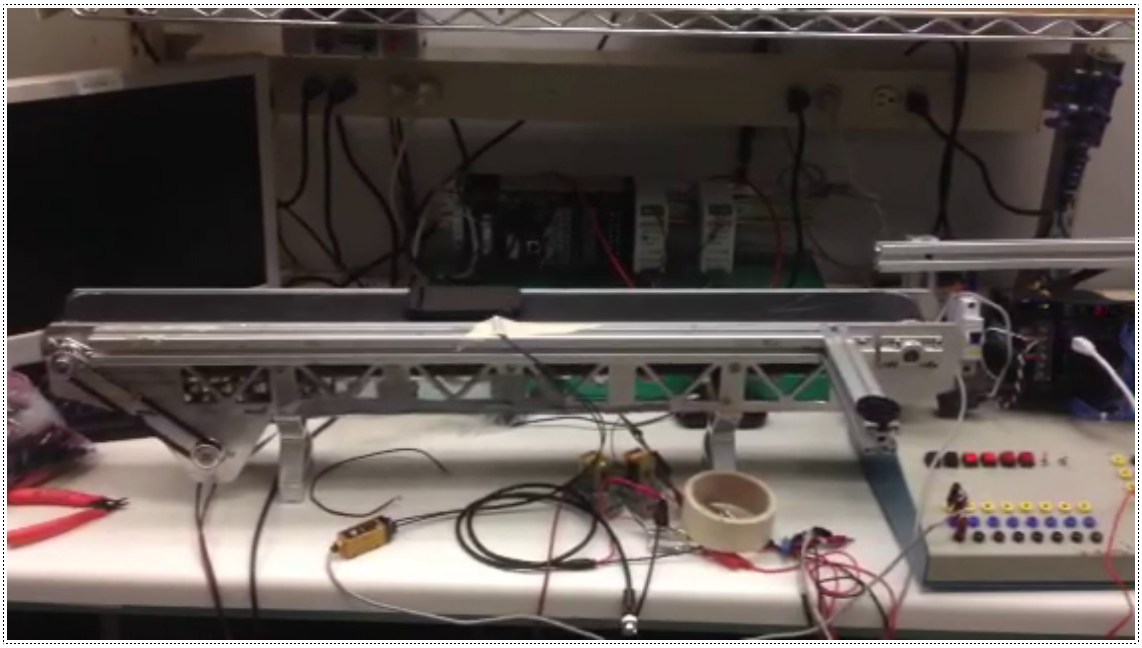


*Illustration 3: At this point the speed has reached its peak value. Once the object passes the sensor in the middle, a timer is set and the conveyor will begin to decelerate.*

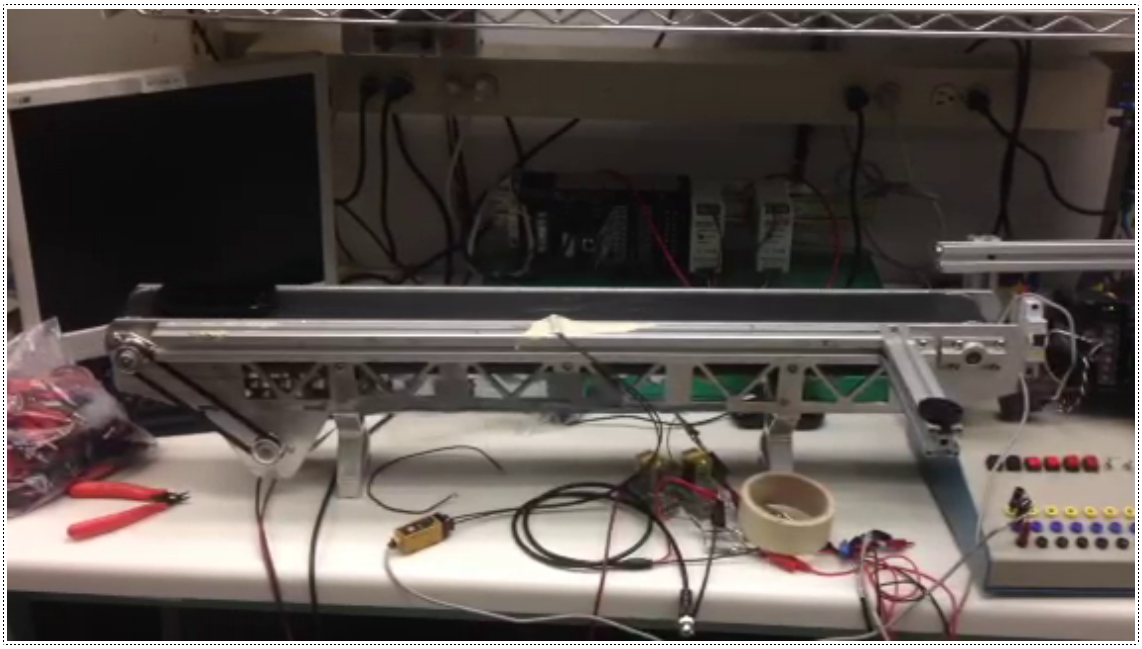


*Illustration 4: As timer value is exhausted and the speed has been slowly lowered, the conveyor belt automatically reverses direction (to the right). A waiting period can be added to allow processing to take place (for example, painting).*

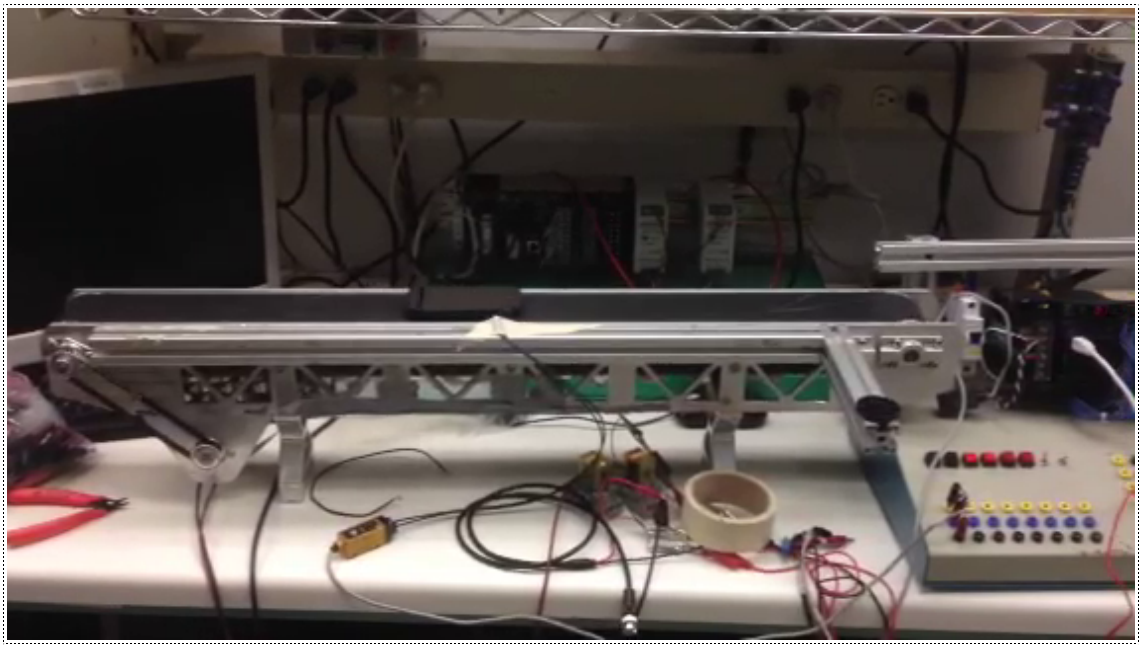


*Illustration 5: The conveyor speeds up, and once the object passes the middle (at maximum speed), a timer is started.*

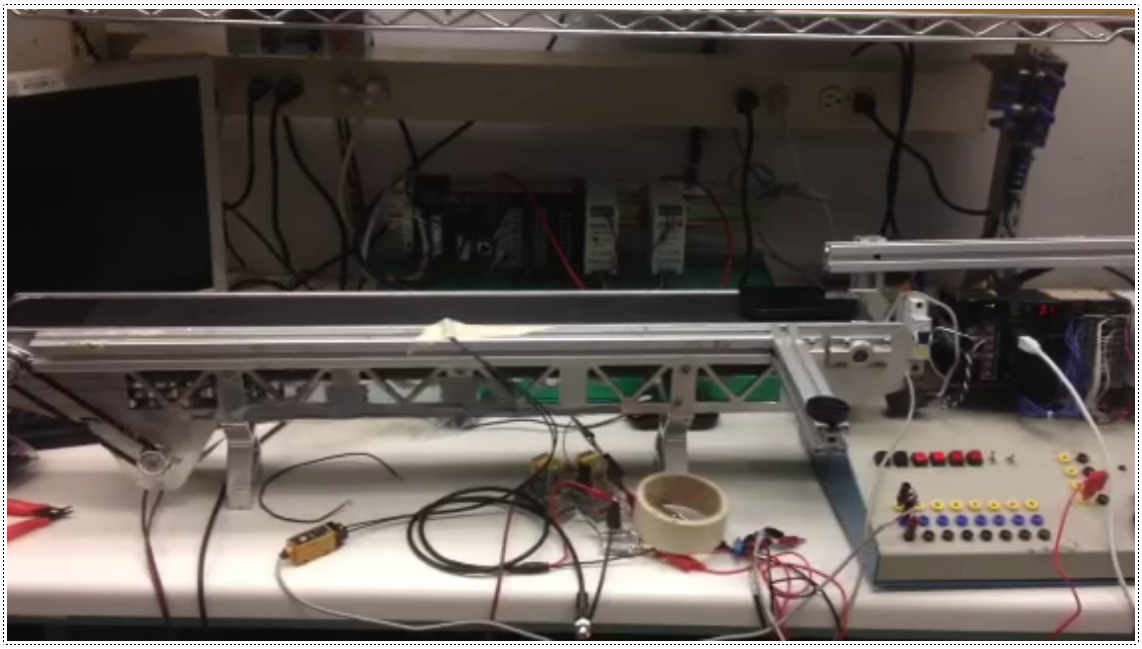


*Illustration 6: As the object reaches its destination, the conveyor slows down; once the object reaches the end, the timer halts the conveyor. This process may now be repeated.*