Super Simple Stepper Move Code.

```
#include <Stepper.h>

// Set up stepper motor controller.
// 100 steps per revolution, and A0-A3 pins
Stepper StepperOne(100, A0, A1, A2, A3);
int StepInput = 0; // Variable to hold incoming numbers.

unsigned long Timer;

// put your setup code here, to run once:
void setup()
{
	Timer = millis();  // LED flashing timer.
	StepperOne.setSpeed(20); // Set Stepper to 20 RPM
	Serial.begin(115200);    // Set up serial port
	pinMode(12, INPUT); // Limits indicator
}

void loop()
{
	// LED Flashing Timer.
	if (millis() - Timer >= 500)
	{
		// Toggle LED.
		if (digitalRead(13))
			digitalWrite(13, LOW);
		else
			digitalWrite(13, HIGH);
		Timer += 500; // Update Timer.
	}

	// Check for incoming serial data.
	if (Serial.available())
	{
		char InChar = Serial.read(); // Read in serial data.

		// Based on character do a certain operation
		if (isDigit(InChar))  // Digit is added into number.
			StepInput = StepInput * 10 + InChar - '0';
		else if (InChar == 'F') // Forward command
		{
			Serial.print("Forward ");
			Serial.print(StepInput);
			Serial.println(" Steps");
			StepperOne.step(StepInput); // Move stepper forward.
			StepInput = 0; // Reset input reading variable.
		}
		else if (InChar == 'B') // Backward command
		{
			Serial.print("Backward ");
			Serial.print(StepInput);
			Serial.println(" Steps");
			StepperOne.step(-StepInput); // Move Stepper backward.
			StepInput = 0; // Reset input reading variable.
		}
		else
			StepInput = 0;
	} // End of Serial available if

} // End of loop
```

In order to fix the blocking problem and to allow the addition of a variable step time, a software timer will be used to sequence the steps. We can also add in a limiting switch, we will employ an optical interrupter for this, the image shows the physical form.

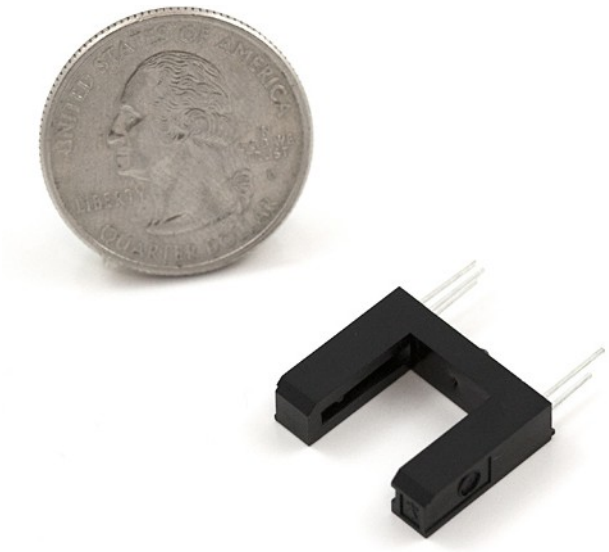

Figure Stepper-1. Physical Form of Opto-Interrupter

The schematic of the driving circuits for the opto-interrupter is shown here.

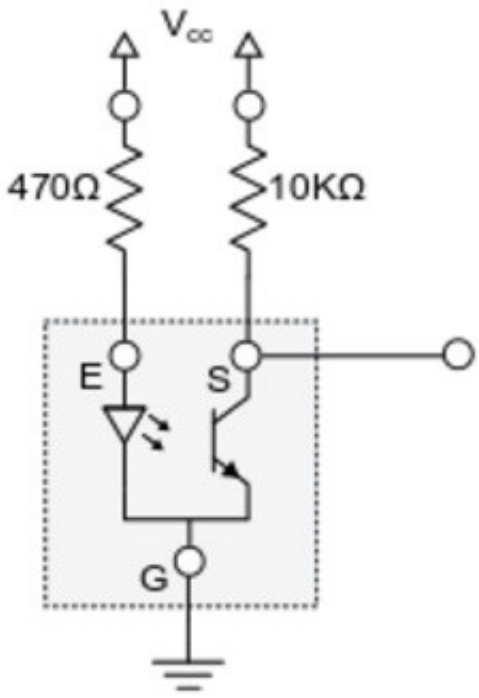

Figure Stepper-2. Driver Circuitry for Opto-Interrupter.

Code for Servo Demo, employing a limit detector.

```
#include <Stepper.h>

// Set up stepper motor controller.
Stepper StepperOne(100, A0, A1, A2, A3);

// Stepper Motor motion parameters
int StepperLocation = 100;    // Current location Arbitrary
int StepperDestination = 100; // Desired Destination
int StepInput, StepperMoving = 0; // Variables for input and feedback.
int Reset = 0;

// function to handle moving stepper forward.
void StepperMove()
{
	// if destination is lower than location
	if (StepperLocation > StepperDestination)
	{
		if (digitalRead(12) == 0 // if limit indicator is active
			&& !Reset)           // but not trying to correct.
		{
			// Set Destination as two steps back.
			StepperDestination = StepperLocation + 3;
			Reset = 1;
		}
		else // Take a negative step.
		{
			StepperOne.step(-1);
			StepperLocation--;
		}
	}
	else if (StepperLocation < StepperDestination) // if opposite
	{
		if (digitalRead(12) == 0  // if limit indicator is active
			&& !Reset)            // but not trying to correct.
		{
			// Set Destination as two steps back.
			StepperDestination = StepperLocation - 3;
			Reset = 1;
			//StepperOne.step(-2);
		}
		else // Take a positive step.
		{
			StepperOne.step(1);
			StepperLocation++;
		}
	}
	else // if we are at our destination
	{
		if (StepperMoving)
		{
			Serial.println("At Destination");
			StepperMoving = 0;
			Reset = 0;
		} // End of feedback if

	} // End of Pos. Neg. at destination if
} // End of StepperMove

#define STEP_INTERVAL 20
unsigned long StepTimer;
unsigned long Timer;
unsigned long CurrentStepInterval = 200; // Initial step time.
```

```
// put your setup code here, to run once:
void setup()
{
	StepTimer = millis();    // Timer handling the Steppr.
	Timer = millis();
	StepperOne.setSpeed(60);
	Serial.begin(115200);
	pinMode(12, INPUT); // Limits indicator
}

void loop()
{
	// timer used to phase out the steps on the stepper motor.
	if (millis() - StepTimer >= CurrentStepInterval)
	{
		StepTimer = millis(); // Update timer.
		StepperMove(); // Update stepper.

		// Adjust Stepper time interval, ramping up ,then down
		if (abs(StepperLocation - StepperDestination) > 10)
		{
			if (CurrentStepInterval > 20) // Provided we are not moving to fast.
			{
				// Speed up as we get to moving.
				CurrentStepInterval -= 20;
			}
		}
		else // We are getting close to our destination
		{
			// Make interval larger until it is 200 milliseconds.
			if (CurrentStepInterval < 200)
				CurrentStepInterval += 20;
		}

	} // End of Timer if.

	// LED Flashing Timer.
	if (millis() - Timer >= 500)
	{
		// Toggle LED.
		if (digitalRead(13))
			digitalWrite(13, LOW);
		else
			digitalWrite(13, HIGH);
		Timer += 500; // Update Timer.
	}

	// Check for incoming serial data.
	if (Serial.available())
	{
		char InChar = Serial.read(); // Read in serial data.

		// Based on character do a certain operation
		if (isDigit(InChar))  // Digit is added into number.
			StepInput = StepInput * 10 + InChar - '0';
		else if (InChar == 'F') // Forward command
		{
			Serial.print("Forward ");
			Serial.print(StepInput);
			Serial.println(" Steps");
			StepperDestination += StepInput;  // add step into location
			StepInput = 0; // Reset input reading variable.
			StepTimer = millis();
			CurrentStepInterval = 200; // Reset Step Interval
			StepperMoving = 1;  // Indicate stepper is moving.
		}
```

```
            else if (InChar == 'B') // Backward command
            {
                  Serial.print("Backward ");
                  Serial.print(StepInput);
                  Serial.println(" Steps");
                  StepperDestination -= StepInput;  // subtract step from location.
                  StepInput = 0; // Reset input reading variable.
                  StepTimer = millis();
                  CurrentStepInterval = 200; // Reset Step Interval
                  StepperMoving = 1;  // Indicate stepper is moving.
            }
            else
                  StepInput = 0;
      } // End of Serial available if

} // End of loop
```