```
#define BUFFER_SIZE 32  //Array size
#include <Wire.h>

int M1dirpin = 7;
int M1steppin = 6;
int M1en = 8;
int M2en = 12;


void Moto_RUN()
{
  Serial.println("Motor: RUN\n"); // send string to serial bus
}

//Motor Initialisation and Setup
void setup() {
  Serial.begin(9600);           // set up Serial library at 9600 bps
  Serial.println("Stepper start");  // send string to indicate the code has begun

 pinMode(M1dirpin, OUTPUT);
 pinMode(M1steppin,OUTPUT);
 pinMode(M1en,OUTPUT);

 digitalWrite(M1en,LOW); //Enable (HIGH: Disable, LOW: Enable)
 digitalWrite(M2en,HIGH);  // Disable

}
void loop()
{
}
void serialEvent()
{
 static char Buffer[BUFFER_SIZE] = "";  // Buffer array
 static char temp[BUFFER_SIZE] = "";    // Temporary array
 char c;
while(Serial.available())
 {
  c = processCharInput(Buffer, Serial.read());  // Read input from serial buffer
  if (c == '\n')
  {
    // Buffer contents fully recieved once terminal byte is read
    if (strcmp("ON", Buffer) == 0)
    {
         // Proceed for 'ON' command to turn motor on
          do
```

```
          {
         Serial.println("Motor: ");
         // Code for DFRobot Stepper shield to drive motor
          int j;
         delayMicroseconds(2);
         digitalWrite(M1dirpin, HIGH);  // Direction setting
         for(j=0; j<=5000;j++)
          {
            // Do not need to be changed
           digitalWrite(M1steppin, LOW);
           delayMicroseconds(2);
           digitalWrite(M1steppin,HIGH);
            delay(1);

        processCharInput(temp, Serial.read());  // Read serial buffer each loop
          }
         Serial.println("ON.\n");  // Motor status is ON
         }while (strcmp("OFF", temp) != 0); // Stop motor when OFF is read from
serial buffer
       Serial.println("STOP\n");  // If 'OFF' chosen, STOP is printed to indicate
motor has stopped
     }
     if (strcmp("RUN", Buffer) == 0)
     {
       // Motor is recieving RUN string in buffer, labview will proceed to send
'ON' command.
        Moto_RUN();
     }
     Buffer[0] = 0;
   }
 }
 delay(10);
}

char processCharInput(char* Buffer, const char c)
{
 //Store the character in the input buffer
 if (c >= 32 && c <= 126) //Ignore control characters and special ascii characters
 {
   if (strlen(Buffer) < BUFFER_SIZE)
   {
     strncat(Buffer, &c, 1);   //Add it to the buffer
   }
   else
   {
     return '\n';
```

```
   }
 }
 else if ((c == 8 || c == 127) && Buffer[0] != 0) //Backspace
 {

  Buffer[strlen(Buffer)-1] = 0;
 }

 return c;
}
```