Demonstrating Transit Method for Exoplanet Detection using Arduino

*Project Report for *Indian Institute of Technology, Bombay* - PH 435: Microprocessors Lab (2023)

Navdha
Department of Physics
210260035

Pal Aggarwal
Department of Physics
210260038

***Abstract*—The transit method is a well-known technique used for detecting exoplanets in astronomy. In this project, we have built a model demonstrating the same with the help of Arduino. The project consists of a star-planet model aligned edge-on with a light sensor interfaced with Arduino. Light detected by the sensor is used to generate light curves with regular dips indicating the existence of a body orbiting the central source. These light curves are then analyzed to determine and verify the system parameters.**

I. Introduction

Most known exoplanets have been discovered using the transit method. A transit occurs when a planet passes between a star and its observer.

Transits reveal an exoplanet not because we directly see it from many light-years away, but because the planet passing in front of its star ever so slightly dims its light. This dimming can be seen in light curves – graphs showing light received over a period of time. When the exoplanet passes in front of the star, the light curve will show a dip in brightness. Such dips if seen periodically indicate the presence of a celestial object orbiting the star.

This method only works for star-planet systems that have orbits aligned in such a way that, as seen by the observer, the planet travels between the observer and the star and temporarily blocks some of the light from the star once every orbit. For our model, we have chosen the star-planet system to be edge-on with respect to the observer (i.e., sensor) to ensure maximum possible dip in brightness.

Transits can help determine a variety of different exoplanet characteristics. The size of the exoplanet's orbit can be calculated from how long it takes to orbit once (the period), and the size of the planet itself can be calculated based on how much the star's brightness lowered.

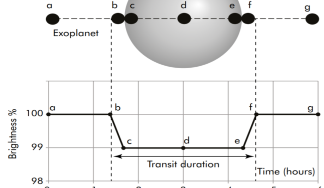

Fig. 1: Transit Method for Exoplanet Detection

II. Hardware

The primary hardware components used for the project include

- Grove Light Sensor
- Arduino Uno
- 12 V DC Motor

A. Grove Light Sensor

The Grove - Light sensor integrates a photo-resistor (light dependent resistor) to detect the intensity of light. The resistance of photo-resistor decreases when the intensity of light increases. A dual OpAmp chip LM358 on board produces voltage corresponding to intensity of light(i.e. based on resistance value). The output signal is analog value, the brighter the light is, the larger the value.

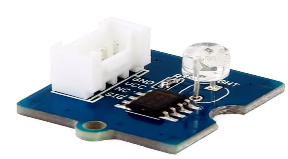

Fig. 2: Grove Light Sensor

The specifications of the sensor are as follows:

Property	Value
Operating voltage	3-5 V
Operating current	0.5-3 mA
Response time	20-30 milliseconds
Sampling rate	30-50 samples/sec
Peak wavelength	540 nm
Weight	4g

III. Method

The procedure was divided into two parts -

1) Constructing the star-planet model
2) Interfacing the light-sensor with arduino
3) Analyzing the obtained data to obtain the angular frequency of the planet

For the former, we have used a 12V DC motor, operated at 0.5-1V to ensure an angular velocity low enough for the light-sensor to be able to sample the low-intensity points. The time period is set to about $\Delta t \approx 20s$, leading to an angular velocity of $\frac{2\pi}{\Delta t} \approx 0.314$ rad/s. The entire setup is enclosed within a dark chamber to avoid noise from ambient light, since the intensity of the source (the sun) is lower than the noise. Any greater intensity would have led to saturation, giving only δ-function like dips, which do not capture the entire transit. Within the box is also our light-sensor, which has been placed at an appropriate position to be able to detect the planet passing in front of the sun edge-on.
The sensor collects the intensity values, which are converted to an analog voltage between 0-5V. This is read by the pin $A3$ (analog input pin) as a value between 0-1023, 0 corresponding to 0V and 1023 corresponding to 5V. We then map these values to lie between 0-500 (called mapped voltage levels from hereon), which are plotted against the time axis to obtain light curves. The baud rate may be adjusted to scale the time-axis up or down.

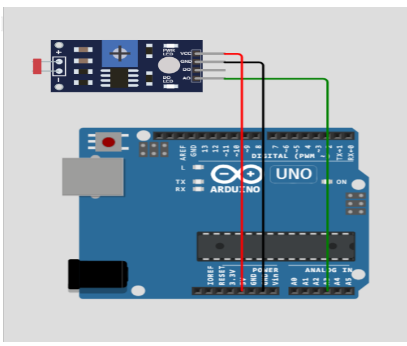

Fig. 3: Circuit Diagram for Sensor Interfacing with Arduino

We use the TimerOne library provided by Arduino IDE to calculate the angular-frequency from the observed light-curves. A threshold value ranging between 15-25 is set to register a dip. This value primarily depends on the position of our sensor with respect to the central source. An interrupt is generated every 1ms to check if the mapped voltage level has become less than the threshold, i.e., a transition from the state with it being above the threshold to being below. If it is, the timer is set to 0, and it counts up every millisecond, until another state transition is registered. This is when the timer count is noted and the timer is reset, so that the process starts all over again. This gives us the time period between two successive dips up to millisecond order. The precision of the method may be increased by decreasing the time interval over which interrupts are generated.

IV. Working and Results

As expected, we see periodic dips in the light curve plotted using Serial Plotter. One such plot is shown in Figure 3:

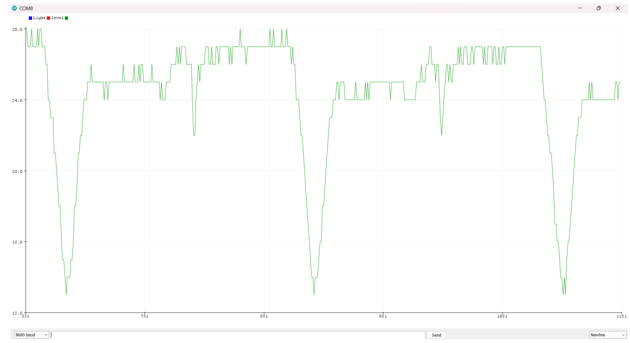

Fig. 4: Lightcurve with periodic dips

The periodic dips are clearly evident in the above light curve.

Once we obtain the lightcurve, we can next use the periodicity to find the period of rotation of the planet around the star which is done as described in the last section.

One such set of readings obtained is shown below:

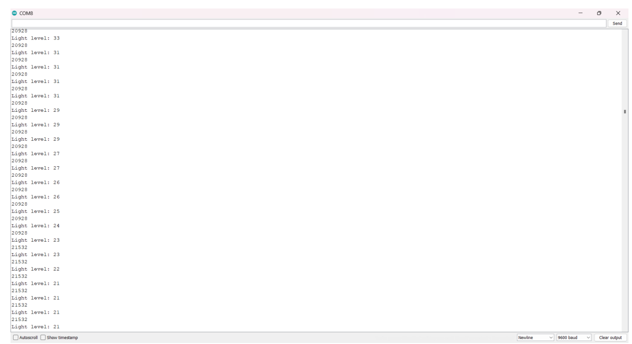

Fig. 5: Time period from periodicity

(As we can see, the time period is actually increasing a bit which indicates the slowing of the motor. One possible reason might be the coiling of the LED lights around the motor causing it to slow down. This change occurs only after a new dip is detected, as shown by the change occurring once the light level dips below 25 which is set as the current threshold.)

V. Conclusion

In this project, we demonstrate a simple model of exoplanet detection using the mthod of transits using Arduino. We also obtain the time-period from the periodic dips using the functionalities provided by arduino. This report presents an elementary introduction to transit method and the methods employed to build the project, along with a detialed overview of the results obtained. This project may be used as an introductory guide to transit method, and several additions may be included to replicate an actual star-planet system with high accuracy.

VI. Author's contribution

After figuring out sensor interfacing with Arduino, both of us worked on building the star-planet model. Once it was assembled, we wrote the code for finding the time period of revolution. Both of us have worked together at all stages of the project.

VII. Acknolwdgements

We would like to thank Prof. Pramod Kumar for giving us the opportunity work on this engaging topic. We are also grateful to Mr. Nitin Pawar, Ms. Swapnali Gharat, Ms. Snehal and Mr. Nilesh for their guidance and support throughout the project phase. Finally, we would like to take this opportunity to express our gratitude to our mentor, Advait Mehla, for his timely inputs which helped in making this project successful.

VIII. References

- https://wiki.seeedstudio.com/Grove-Light_Sensor/
- https://exoplanets.nasa.gov/faq/31/whats-a-transit/
- https://lco.global/spacebook/exoplanets/transit-method/#:~:text=This%20method%20only%20works%20for,the%20star%20once%20every%20orbit
- https://www.arduino.cc/reference/en/libraries/timerone/

IX. Appendix

A. Arduino code for the project

```
#include<TimerOne.h>
int light_sensor = A3;
volatile bool prevState = HIGH
volatile bool nextState = HIGH;
volatile int count = 0;
volatile int timer_1 = 0, timer_2 = 0;
volatile unsigned long int time_period = 0;

void setup() {
  Timer1.initialize(1000);
  //initialize timerl to lms, will generate an
  interrupt every lms
  Timer1.attachInterrupt(time_tracker);
  Serial.begin(9600);
  //begin Serial Communication
}

void loop() {
  int raw_light = analogRead(light_sensor);
  // read the raw value from light sensor
  pin (A3)
  int light = map(raw_light,0,1023,0,500);
  // map the value from 0, 1823 to 0, 500
  if (light < 25) {nextState = LOW;}
  // if voltage goes beyound a certain
  threshold (i.e., the intensity
  drops below a certain threshold)
  else{nextState = HIGH; }
  Serial.print("Light level: ");
  Serial.println(light);
  // print the light value in
  Serial Monitor
  Serial.print("Timer period (in ms): ");
  Serial.println(time_period);
  delay(100);
}

void time_tracker() {
  if (prevState==HIGH && nextState==LOW)
  { // rising edge detected
    if (count == 0) {//first rising edge
      count++;
      time_period = timer_2;
//read time difference between
      two consecutive dips
      timer_1 = 0;  //start tracking time
      timer_2 = 0;  //reset time
      //timer_2 = 0;

  }
    else if (count == 1){ //2nd rising
    edge detected
    //reset timer and count
      time_period = timer_1;
      //read time difference between
      two consecutive dips
      timer_1 = 0; //reset time
      timer_2 = 0; //start tracking time
      count = 0;  //reset count

      }
    }
    else{
    //update timers
    timer_1++;
    timer_2++;
}
    //transfer new state to prevState
    prevState = nextState;
    }
```

B. Video of the working demo

A video of the working demo, with the planet revolving around the star and the light curves being generated in real time can be found here:
https://drive.google.com/file/d/1FaufGqzs8wnMvBx9E782EhxV5vf_nNgp/view?usp=drivesdk