Solar room heating automation

June 9, 2025

Abstract

The Solar Trapper is a project to find a cheap and simple way to heat houses for those who are unable to afford heating. The solution to this issue is to create a system that automatically closes the curtains when the sun no longer heats the house, insulating it from loosing its heat and therefore saving money on heating.

This project is free and open source [1].

Contents

1 Introduction

In Britain, many people face the issue of freezing due to the inability to heat. Indoor heating is a massive energy consumer in the UK, responsible for about 20% of greenhouse gas emissions[2]. This not only contributes to climate change but also leads to unsustainable expenses for households who need to heat during winter.

The aim of this project is to develop a device that permits people to make more use of solar energy to heat their rooms, while still keeping the cost of hardware and installing the system low.

The potential of using solar energy is significant, about $1.3KW/m^2$ [3]. Even if the majority of the energy is lost to its surroundings and only $500W/m^2$ are captured, this may save up to $5Kwh$, or the equivalent of 1 Kg of firewood for each m^2 of a south-facing window.

1.1 Photovoltaics for room heating

An obvious choice to start with would be photovoltaic panels, using the generated electricity to heat the water used in the heaters, but a cursory market study of available components indicates that an adequately sized photovoltaic system (roughly 2kW of power) would cost roughly £4000[4], well above the budget of typical persons in need of additional heating and with an unclear amortisation schedule. Therefore, we rule out using photovoltaics for electric heating.

1.2 Air collectors for room heating

Another method that was considered was using Air collectors for room heating, which work by using solar energy to heat the air, which is then circulated into a room. They are simple and robust as well as providing fresh air to the room, ensuring the house does not grow moist when the owner is away for longer periods. However, they are very expensive at roughly £2200 for 2kW heating output[5]. Also, they are difficult to install and often only ideal for a permanent home, so those renting a house - generally those who require it the most - would be unable to make use of it.

1.3 Window insulation

A very cost effective way of keeping the temperature up in a room is to open curtains when the sun is heating the space through the window, and then close the curtains when the temperature starts to fall again in the evening. The issue with this is that many people are away at work during the optimal time for closing the curtains.

For this reason, we decided that the most prudent course of action for this project is to design a system that automatically closes the curtains at the optimal time, letting the sun in only while it still effectively heats.

In addition, this system is very simple and cheap, which would allow those who need this extra insulation to afford it.

The system we designed closes the curtains when the sun goes away or the temperature in the room starts to drop, so as to keep the solar heat inside and reduce heating costs.

It has a detector that can sense very bright light (i.e., the sun), a sensor to measure the temperature accurately and an actuator to release a curtain closing mechanism, which is triggered when the sun has vanished for a certain time or becomes too weak to effectivly heat the room. The system does not respond immediately, so that wrong measurements or reappearance of the sun do not result in reduced energy harvesting.

For cost reasons, the system does not support opening curtains automatically, as originally considered. Instead, the user needs to reset these manually. Closing works by releasing a weight with a servo arm. The potential energy of the weight moves the curtain via simple pulleys.

Subsequent to closing action, the locking mechanism moves back into the original position for convenience in resetting the weight and curtain to harvest solar energy.

2 Project Plan

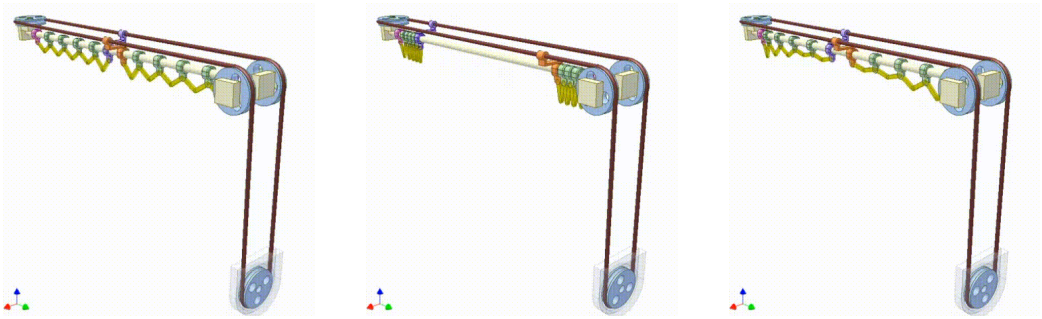
2.1 Overview

This paragraph outlines steps in implementing the project described in the previous section.

Table 1 lists the various steps required in this project and their deadlines, along with the dates for other competitions this project was submitted in.

Table 1: Project timeline

Work package	Due date
Formulate problem	8 January 2025
Research suitable sensors	13 January 2025
Log data provided by sensors	20 February 2025
Plot data	22 March 2025
Develop algorithm for curtain closing time	23 March 2025
Build hardware	23 March 2025
Create video presentation	24 March 2025
Submit to Raspberry Pi competion to get feedback	24 March 2025
Write project report, submit to CREST	31 June 2025



(a) Closed (b) Open (c) Closed again

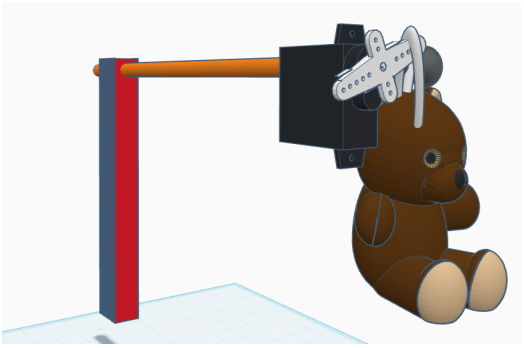
Figure 1: Mechanism for opening and closing curtains, requiring several pulleys, connectors and a stepper motor [6]. DC motor attached to a pulley spins when activated, moving the curtains attached to the pulley system.

2.2 Curtain Closing mechanism

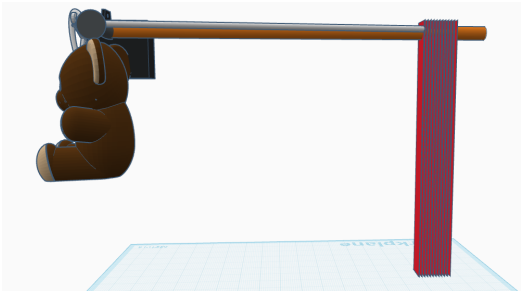
One of the main challenges we faced was to design a mechanism that could close the curtains in a cheap and reliable way. An example of such a rather complex and costly system is shown in Figure 1. We designed a similar system on a large curtain.

While seemingly simple at the start, the design quickly became less than ideal, as it would require additional sensors to signal the motor to stop, and the resistance between the curtain and the railing was so great that the pulley connected to the motor could not overcame it. Both of these problems were solvable, but the cost would be increased significantly, and so we designed a new system:

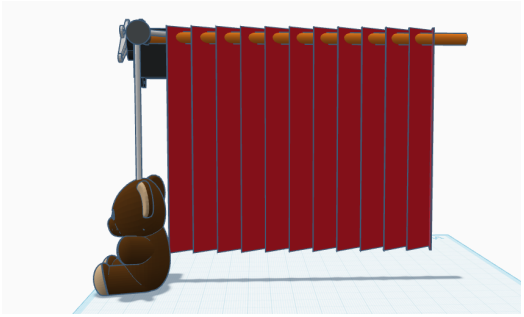
A weight would be connected to the curtain, to be released by a servo. The weight would fall and provide the energy to close the curtain. The user



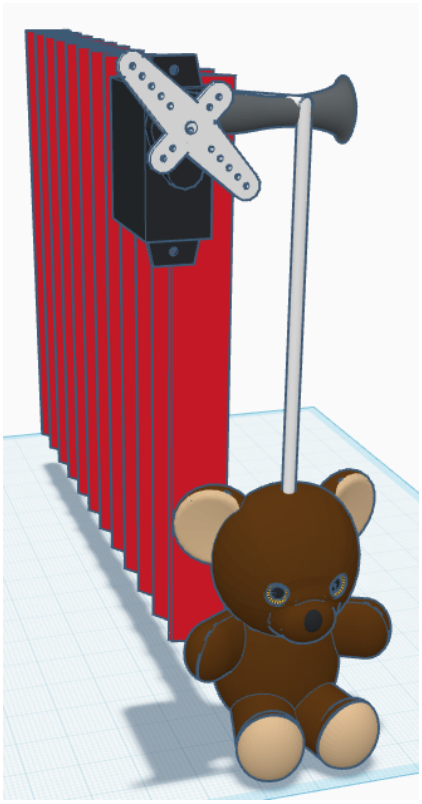
(a) Open



(b) Open



(c) Closed



(d) Closed

Figure 2: Curtain system: Open and closed state. Teddy bear represents weight.

has to reset the curtain when the sun is about to heat the room again in the next morning. Although this system is only able to close curtains, it provides the cheap and simple option we needed, so we settled onto this version.

To reduce friction, we used fishing wire and a pulley. This allowed us to to roll the wire around the pulley several times, ensuring that it would not slip off. Our newtonmetre recorded that we needed roughly 0.4 N, the equivalent of an average size teddy bear. It worked perfectly first time, but for the pulley to hold, we required super glue as hot glue wasn't strong enough. Afterwards, the servo was attatched under it, to be connected to a loop at the bottom.

2.3 Sensor selection criteria

In order for the system to accurately judge the optimal time to close the curtain, external sensors would be required. Selection criteria for sensors include:

- Cost
- Robustness
- Measurement accuracy
- Simple to integrate
- Availability

The data it requires would be the amount of energy going into the room and the energy going out. Thus, our options were:

- A temperature sensor - measure the temperature to determine when the temperature drops and ensure the Photoresistor does not act when the sun goes behind a cloud.
- A UV detector - measure the strength of the suns rays.
- A Photoresistor (A light sensor) - measure how bright it is, if the sun has gone down and to get a quicker reaction than the temperature sensor.
- A Photo Diode (A faster-responding light sensor) - as the Photoresistor.

Table 2: Sensor trade study. 5 = perfect, o = terrible

Sensor	Price	Availability	Accuracy	Simplicity	Total
Photo Resistor	4	3	4	2	13
Temperature Sensor	4	4	4	5	17
Photo Diode	1	4	5	3	13
UV detector	1	3	2	1	7

As seen in the trade study in Table 2, the Temperature sensor, the Photo Diode and the Photo Resistor scored the highest points. While the Photo Diode would be very accurate, it is unneccessarily expensive when a Photoresistor would do just fine. A UV sensor has a high price and needs to be placed outside to work effectively, which would make the process of installing it too complicated. Therefore, we chose a Temperature Sensor and a Photoresistor as the sensors for the curtain system.

2.4 Equipment

Table 3 lists the parts that were required while building this project. This shows that this design is a very econimic solution, as most of the costs come from the model parts. On an actual curtain, the cost would likely be an estimated £30, affordable by most people.

Table 3: Equipment

Parts	Price	Source
LDR Photo Resistor	£ 3.99	Bought by Mr. Chikunga
DS18B20 Temperature Sensor	£ 3.95	Bought by Mr. Chikunga
Servo Motor	£ 3.99	Scavenged from a robot
Arduino Nano	£ 4.99	Scavenged
Micro SD Card	£ 7.99	Bought on Amazon
Salvaged Parts	£ 1	-
Model Stand	18.99	Bought on Amazon
Model Curtain	10.19	Bought on Amazon
Total	£ 55.09	-

3 Project execution

For demonstrations and work in school, the hardware needs to be easily transportable. Hence, rather than a fixed install of a full sized system, we

(a) Open (b) Servo Released (c) Closing (d) Closed

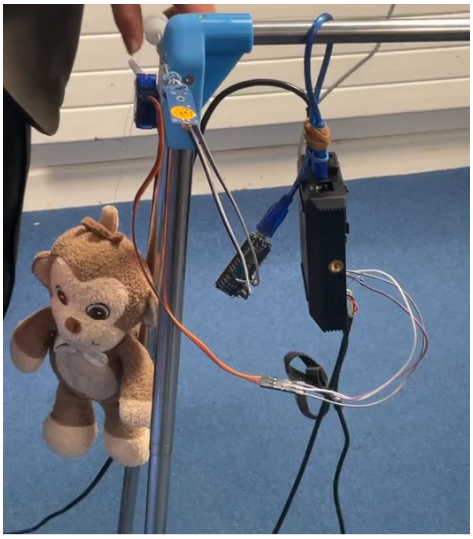
Figure 3: Model system: System is activating after temperature and light levels drop.

decided to build a model. The result is the stand with a model curtain seen in Figure 3.

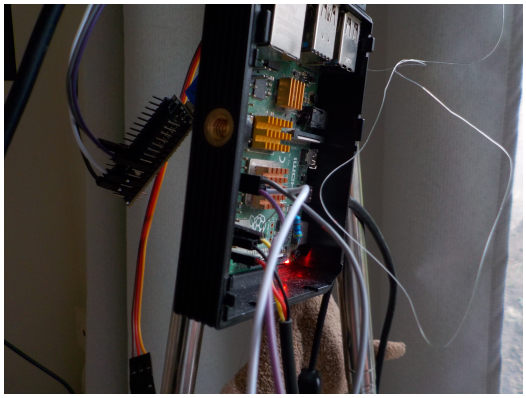
3.1 Hardware Assembly

The following parts and tools were used in assembling the model:

- Servo
- Weight
- Fishing Wire
- Pulley and a suitable Axle, loose enough to spin with relative ease.
- DS18B20 Temperature Sensor
- LDR Photo Resistor (Light Sensor)
- Raspberry Pi
- Arduino Nano
- Micro SD Card
- Wires
- Hot Glue Gun



(a) Release System: Servo, pulley and light sensor are fitted to the blue bracket. Raspberry Pi and Arduino Nano are temporarilly suspended on the rail. The temperature sensor (out of frame) is connected via the black cable pointing downwards. The monkey, if released, pulls the curtain.



(b) Wiring on the RPi: Servo and temperature sensor are connected to the GPIO pins, servo wired in grey, purple and white, temperature sensor wires in red, black and yellow. In the background, the Arduino Nano has the light sensor board connected to read data and provide power.

Figure 4: Close-up of release system components.

3.1.1 Release System

The following steps are to install the system responsible for closing the curtain. Be careful to place the parts in the correct location, to avoid the wire from being misaligned. Also refer to Figure 4.

- Put the pulley on the axle.
- Attach the axle to one end of the curtain railing so that it is perpendicular to the movement of the curtain. Glue using 2 component super glue, then coat it with a layer of hot glue for stability. The pulley should be close to but not touching the railing.
- Use fishing wire to connect the end of curtain to the weight through the pulley, then create a loop further along to hook the servo to.
- Use hot glue to secure the servo motor to the model stand just below, but not in the way of, the pulley.
- Ensure the servo arm can move freely to release the weight.

3.1.2 Wiring

The final steps in the assembly of the system is to wire everything up (also refer to Fig. 5):

- Connect the VCC pin of the DS18B20 to Pin 1 on the Raspberry Pi *(RPi)*.
- Connect the GND pin of the DS18B20 to Pin 6 on the RPi.
- Connect the data pin of the DS18B20 to the GPIO7.
- Connect the VCC on the LDR Light Sensor to 5V on the Arduino Nano.
- Connect the two GND's together.
- Connect the AO (Analog Output) on the LDR to the A5 pin.
- Once the servo is attached, connect the GND (negative) wire to pin 20 on the RPi.
- Attach the VCC (positive) wire to pin 17 on the RPi.
- Connect the sig wire to pin 16 on the RPi

- Once the above steps have been completed, attach the Arduino Nano to the RPi via USB cable. Upload 'LightSensorTest.ino' to the Arduino Nano.

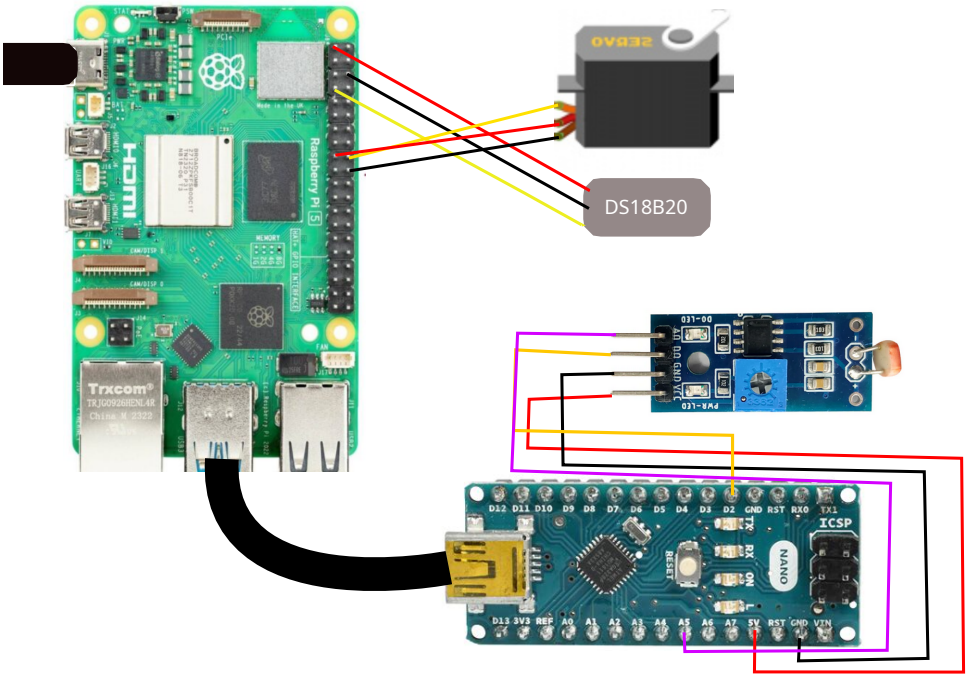


Figure 5: Electronics components of curtain closer. The Raspberry Pi computer directly controls the servo and reads from the digital temperature sensor DS18B20, while the photoresistor board for light level measurements needs to be connected through and Arduino Nano board, which serves as analog to digital converter. Data is transferred from Arduino to Raspberry Pi through a serial connection over USB. The Raspberry Pi is powered using a 2 Amp transformer with USB-C connector.

3.1.3 Software configuration

- Enable the 1-wire protocol on the RPi [7].
- Upload sofware to Arduino board for reading analog light level values and forwarding to Raspberry Pi. Find details in Section 3.2.
- Install the dependencies on the command line interface *(CLI)*: `pip install pandas matplotlib numpy RPi.GPIO`.

3.2 Data logging

For developing and testing algorithms, it is impractical to work with live data. Instead, data should be logged into an easy to read file, and algorithms can then be tested by replaying this file.

The csv log file contains measurements once per second and each measurment needs an accurate time stamp.

Reading the data from the temperature sensor is fairly straightforward, using a one wire protocol[8]. This means that it just requires one data line to communicate with the rasberry pi. Data can then be read from the system file `/sys/bus/w1/devices/28-3fa9d445207e/w1_slave`, depending on the ID stored on the sensor chip.

A bit more work is required to read light levels from the photoresistor circuit: As these are not digital, an Arduino board is used to convert analog voltage readings into digital numbers. These digital numbers are written onto a serial link with the following program to be loaded onto the Arduino board:

```
const byte Pin1 = A5;

void setup(){
  pinMode(Pin1, INPUT);
  Serial.begin(9600);
}
void loop (){
 long int val1 = analogRead(Pin1);
 Serial.println(val1);
 delay(1000);
}
```

The Arduino is connected to the Raspberry Pi with a USB interface, and the values from the Arduino can therefore be read by opening the system file `/dev/ttyUSB0`, once every second.

The following Python listing illustrates the process of reading data from sensors and clock, and writing these into a log file.

3.2.1 Data Plotting

To easily view the data and possibly develop it, it is necessary to plot the data. Figure 6 displays logged data for a range of days in winter of 2025. Plots were generated from the logged data using pandas and matplotlib libraries.

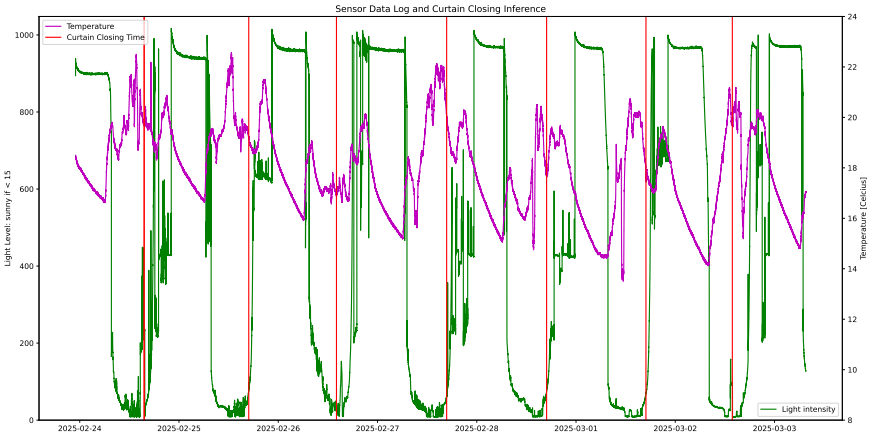


Figure 6: Temperature and light levels logged from 24 February 2025 to 3 March 2025. Ticks on the x-axis indicate midnight of each day. Vertical orange lines indicate for each day the time of conditions for curtain closing being met, i.e., temperature dropping for 30 minutes and light levels too low for effective solar heating. There is one point per second. Note that temperature is rising in early mornings (6-8 am) and evenings (5-9pm) following conventional heating system activation.

4 Development of closing algorithm

The final step was to develop the algorithm that tells the servo when to close the curtains. While we originally attempted to design a system that returns the percentage chance of the closing time being correct, we quickly realised the complexity of the code required, when a simple Boolean output would do. Using Figure 6, we finalised the plot to close the curtains if it is the afternoon, the temperature drops by 0.5°C over a period of 30 minutes and there is not enough sunlight to heat the space, or if it is close to sunset. To check the difference between the current temperature value and the value 30 minutes ago, we used a queue which holds 1800 values, one for each second. Each time it recives a new value, it compares it to the first one, and checks if the temperature has dropped by 0.5°C. Light value increases as it gets darker, so once the light value is more than 15 and the temperature also falls, the curtain closes. In the event that it is simply a very hot day, the curtain system will close around sunset, for which a program exists to compute it[9].

The above expressed in pseudo code is shown below.

```
if time is pm AND
```

```
temperature drops for 30 minutes AND
light level > 15 OR
sunset time reached:
    do: close curtain
```

5 Results and discussion

We tested the system over the period of a week, and the results proved promising. However, in some situations, it may be more benificial to have a more expensive system that can manage itself, capable of both opening, and closing, the curtains to adapt to more unpredictable weather conditions.

6 Future Work

Work in progress includes pulling short range weather forecasts information from OpenMeteo[10] for making better decisions about curtain closing times. We have considered using reinforment learning techniques to automate the system, but these are currently out of scope due to lack of training data and time.

References

[1] The Smart Solar Trap. https://github.com/hermonochy/The-Solar-Trapper

[2] 2023 UK greenhouse gas emissions, provisional figures https://assets.publishing.service.gov.uk/media/6604460f91a320001a82b0fd/uk-greenhouse-gas-emissions-provisional-figures-statistical-release-2023.pdf

[3] Solar Constant https://en.wikipedia.org/wiki/Solar_constant#Apparent_magnitude (retrieved: 1 March 2025)

[4] https://solar-energy-store.co.uk/off-grid-solar-pv-kits/2kw-2000w-solar-panel-pv-kit-system-for-off-grid-hybrid-self-storage-battery-storage-premium/

[5] SolarVenti air collectors: https://www.hess-solar.de

[6] Initial curtain opening and closing mechanism: https://www.youtube.com/shorts/H6Un9TCUDxk

[7] Activating the 1-wire protocol on Raspbian: https://www.raspberrypi-spy.co.uk/2018/02/enable-1-wire-interface-raspberry-pi/, accessed on 27 Feb 2025.

[8] DS18B20 Sensor: https://randomnerdtutorials.com/raspberry-pi-ds18b20-python/

[9] Sunset Equation: https://en.wikipedia.org/wiki/Sunrise$_e quation$

[10] Open Meteo: https://open-meteo.com/, accessed on 26 Feb 2025.