CHAPTER 5

RESULTS AND DISCUSSIONS

The performance of the air conditioning system depends upon the heat transfer capacity of the fluid. R-134a refrigerant is used in the air conditioning system. This refrigerants heat transfer capacity is not good and hence it increases power consumption of the compressor. Due to this limitation, the refrigerant passes through the copper pipe which is surrounded by double pipe heat exchanger unit.

The thermal collector was kept under solar radiation which absorbs and converts solar radiation into useful heat energy, where Al_2O_3 nanofluid was circulated through double pipe heat exchanger to enhance the temperature of refrigerant before it enters into the compressor and thus reduces the power consumption of compressor.

5.1. Analysis of Solar Thermal Collector

The output temperature was studied by circulating water and Al_2O_3/water nanofluid at different flow rates for a period of 5 days. Depending upon on the readings taken, the optimum flow rate was found to be 60L/hr as it possesses the maximum heat transfer rate at an angle of inclination of 25° of the collector in the south direction. The figure 5.1. shows the plot between collector outlet temperature and at the various time intervals that the readings were taken.

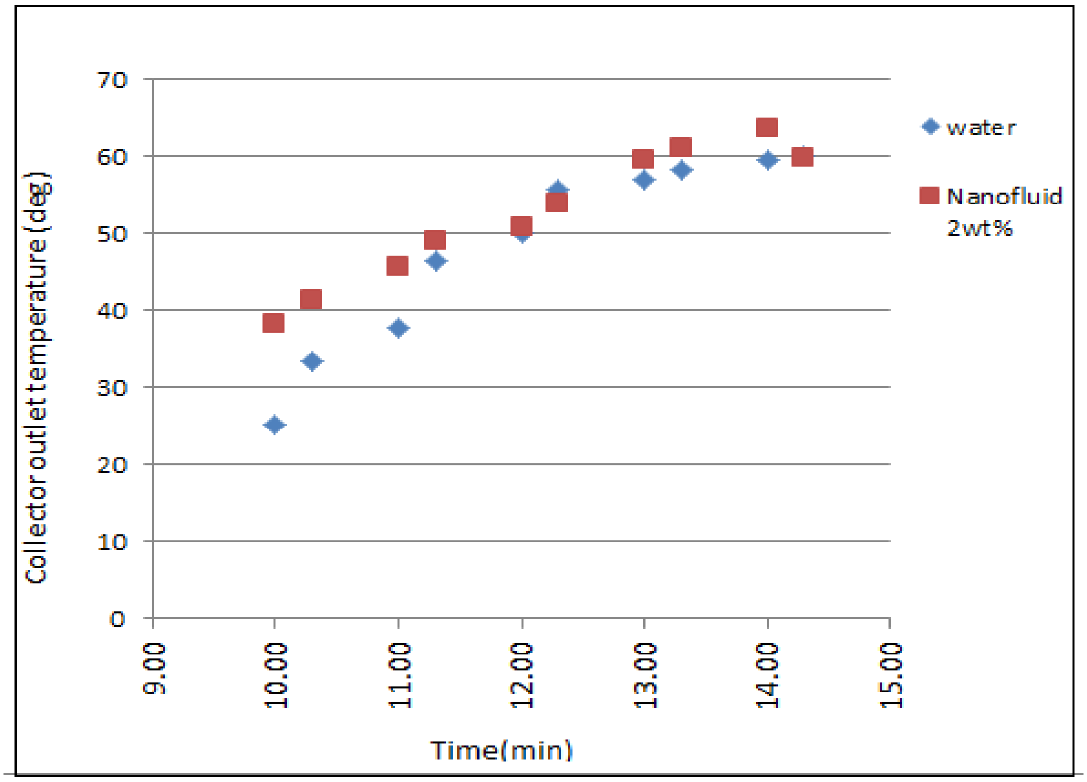


Figure 5.1. Graph between temperature and time of collector

From the above graph, it is clearly seen that the nanofluid of 2 wt % exhibits better heat transfer capacity as compared to that of water.

5.2. Analysis of Solar Air Conditioning System

The COP (Coefficient of Performance) of the air conditioning system was analysed in this section. Refrigerant R-134a is the working fluid in the ideal vapour compression refrigeration cycle. The refrigerant temperatures at various stages of the cycle was measured and their respective pressure-enthalpy values were found by using the R-134a tables. Their values are given in Table 5.2. and the COP is calculated by the formula:

$$COP_{(Refrigeration)} = H_2 - H_1/H_3 - H_2$$

$$= 110-105/119-110$$

$$=0.55$$

Where, H_1 = Enthalpy before evaporator

H_2 = Enthalpy before compressor

H_3 = Enthalpy after compressor

Denotation	Temperature(F)	Pressure(PSI)	Enthalpy(Btu/lb)
H_1	3	8	105
H_2	43	38	110
H_3	112	150	119

Table 5.1. Temperature, Pressure and Enthalpy values of R-134a refrigerant

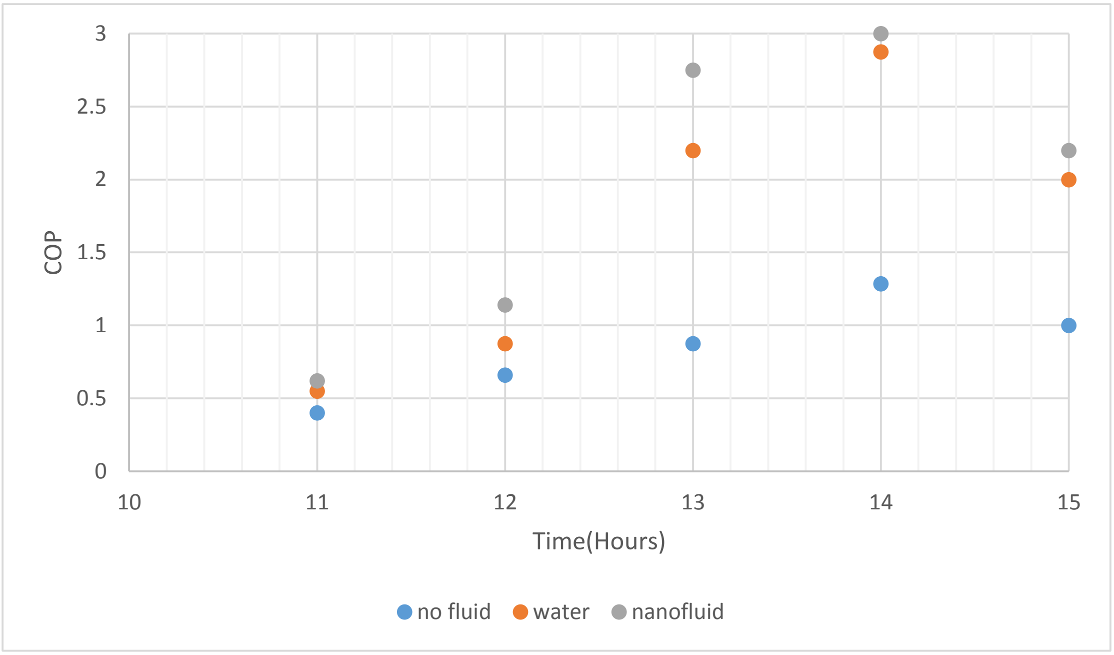


Figure 5.2. Graph between COP and Time

From figure 5.3, its seen that the COP value changes with respect to time and environmental factors such as atmospheric temperature, wind, humidity, etc. The COP calculated by using nanofluid has a higher value than the COP calculated by using water as heat exchanging fluid in the solar assisted air conditioning system.

5.2.2. Power Consumption of Compressor

The running time of the compressor was taken for 4 hours starting from 10:00 to 14:00 with the usage of no fluid, water and nanofluid in the double pipe heat exchanger and their values were plotted in the figures 5.1, 5.2, 5.3 respectively below to depict the overall efficiency of the double pipe heat exchanger. With these values the power consumed by the compressor is calculated.

When no fluid was flowing through the double pipe, the normal air conditioning cycle was operated and the running time was noted and plotted in Figure 5.2. The atmospheric temperature on that day was 36°C.

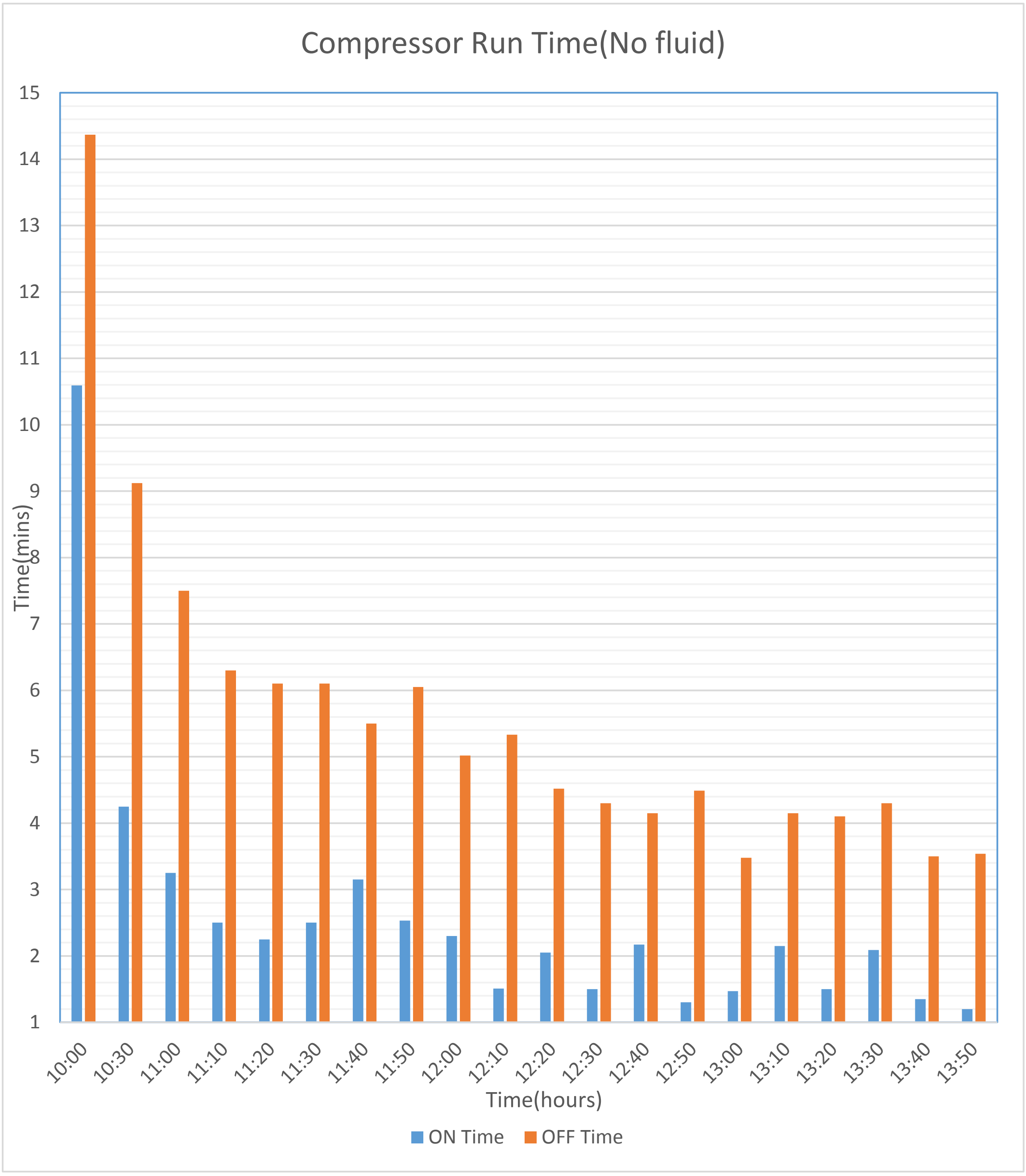


Figure 5.3. Graph showing run time of compressor without fluid

When water was flowing through the double pipe, the normal air conditioning cycle was operated and the running time was noted and plotted in Figure 5.3. The atmospheric temperature on that day was 40°C.

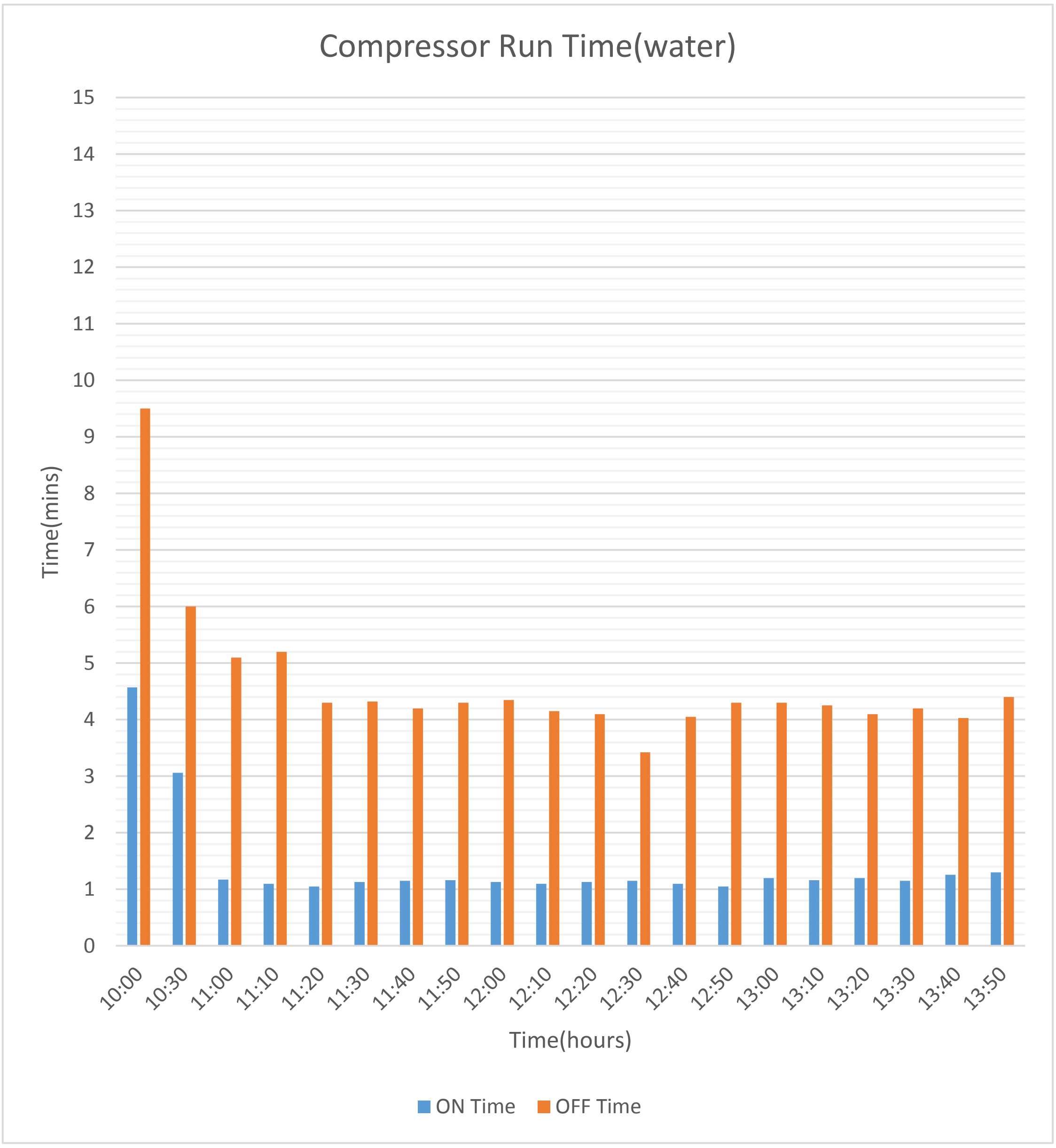


Figure 5.4 Graph showing run time of compressor with water

When water was flowing through the double pipe, the normal air conditioning cycle was operated and the running time was noted and plotted in Figure 5.3. The atmospheric temperature on that day was 38°C.

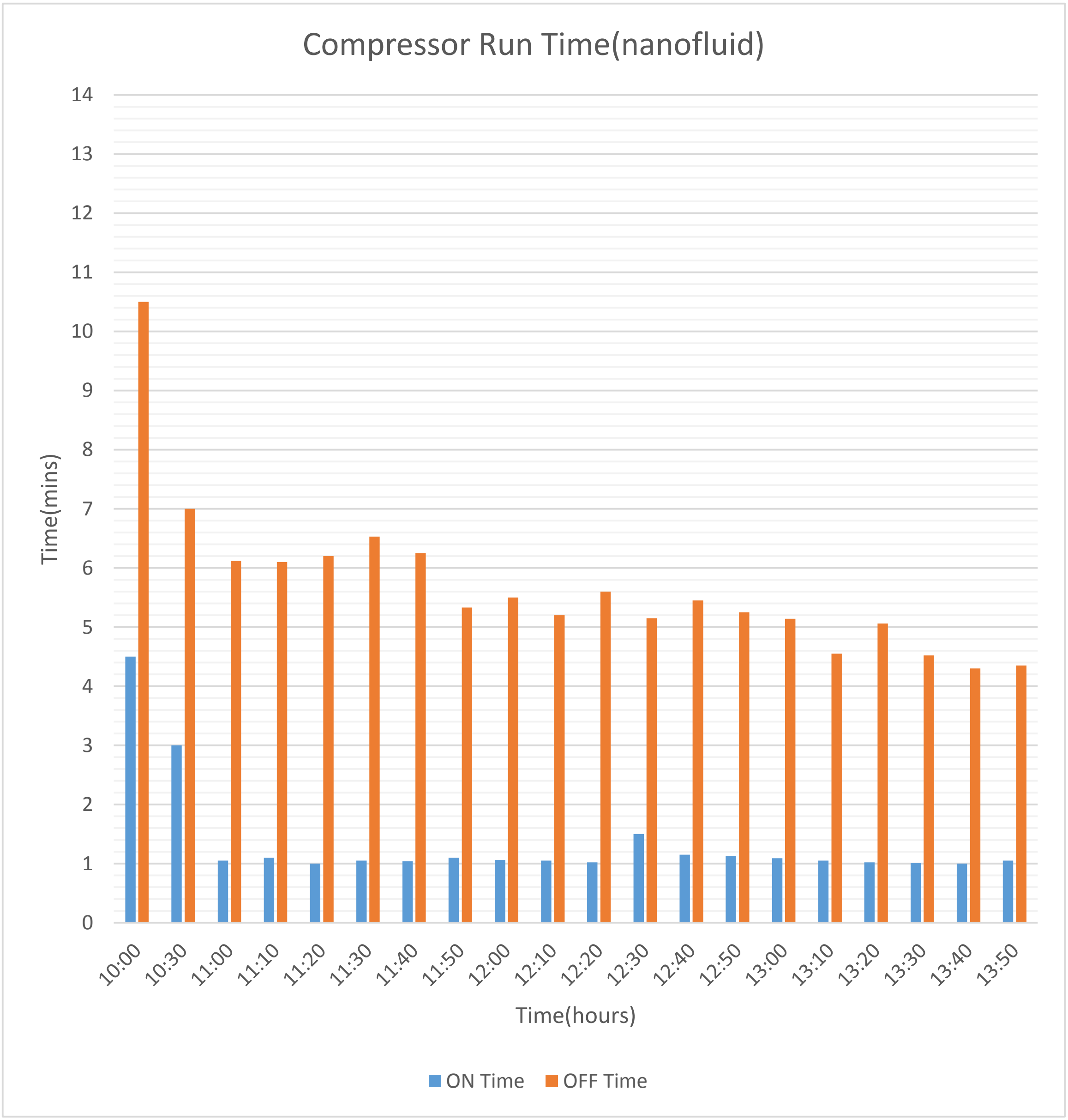


Figure 5.5 Graph showing run time of compressor with nanofluid

From the above graphs, average time has been calculated and the values has been plotted in Table 5.3.

S.no.	**Atmospheric Temperature**	**Fluid**	**ON Time(mins)**	**OFF Time(mins)**
1.	36°C	No fluid	2.58	5.59
2.	40°C	Water	1.35	4.627
3.	38°C	Nanofluid	1.34	5.705

Table 5.2. Comparison of compressor run time

From the above table, it is seen that by using nanofluid as the heat transferring fluid in the double pipe heat exchanger proves to be the most efficient in reducing the average OFF time of the compressor to roughly 1 minute lesser than normal working operation.

CHAPTER 6

CONCLUSIONS AND FUTURE SCOPE

The main aim of this project is to reduce the power consumption of the air conditioning system by adding solar thermal collector which will contribute towards a cleaner environment by inculcating the use of green technology by the following aspects:

- Solar thermal collector absorbs solar radiation and converts it into heat energy and that heat energy is utilised for increasing the temperature of R-134a refrigerant by the exchange in the double pipe which increases the compressor’s OFF time by an average of 1 minute more than the normal operating condition.
- Al_2O_3 nanofluid was prepared using ultrasonic probe sonicator. 2wt% of the Al_2O_3 nanoparticles were suspended in the base fluid-water
- By the various tests carried out with the solar thermal collector, the optimum flow rate was chosen as 60L/hr which was used during the operation of the solar assisted air conditioning system.
- The COP of the system has been enhanced by circulating 2wt% Al_2O_3 nanofluid and the results were compared with the COP calculated by using water which was found to be 6-7% higher on an average.

Some future enhancements which can be made in the system include:

- To be able to run the compressor by using solar panels alone and not with battery supply/electrical supply to make the system completely reliable on natural resources.
- Using different volume concentration of nanofluid to check which will give more efficiency and have better heat transferring capacities within the collector.

REFERENCES

[1] Muhammad Abbas, Rashmi G. Walvekar, Mohammad Taghi Hajibeigy, Farood S. javedi October 2013*"Topology Efficient Air-Condition unit by using nano-refrigerant"*, EURECA 2013.

[2] Juan Carlos Valdez Loaiza, Frank Chaviano Pruzaesky, Jose Alberto Reis Parise. July 2010*"A Numerical Study on the Application of Nanofluids in Refrigeration Systems"*, International Refrigeration and Air Conditioning Conference.

[3] T. Coumaressin and K.Palaniradja, Nov.2014 "*Performance Analysis of a Refrigeration System Using Nano fluid."* International Journal of Advanced Mechanical Engineering.ISSN 2250-3234 Volume.

[4] F.Z.Ferahata, S.Bougoul, M.Medale and C.Abid *"Influence of the air gap layer thickness between the glass cover and the absorber of a solar collector"*, 2012 Tech Science Press FDMP, vol.8, no.3, pp.339-351..

[5] Jingmin ZHANG, Huilan HUANG and Hua ZHANG. *"Experimental Study on Solar Collector with different Thermal Storage Materials"*, International Conference on Power Engineering-2007, October 23-27, 2007, Hangzhon, China

[6] N.R.Avezova, R.R.Avezov, O.S.Ruziev, A.Vakhidov and Sh.I.Suleimanov. *"Longevity Characteristics of flat solar water-heating collectors in hot water supply systems"*, ISSN 0003-701X, Applied Solar Energy, October 2013, Vol.49, No. 1

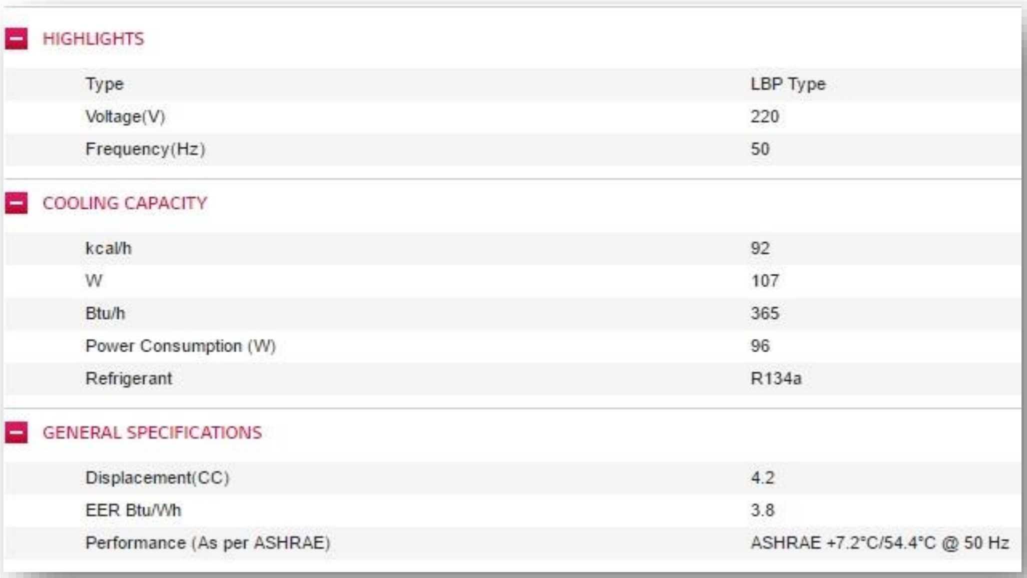
[7] K.Toufek, M.Haddadi and A.Malek, June 2009 *"Experimental study on a new hybrid photovoltaic thermal collector"*, ISSN 0003-701X, Applied Solar Energy, vol.45. Allerton Press.

[8].R..Avezov and N.R.Avezova *“Heat Efficiency of “Translucent Cover—Radiation Heat–Exchange Panel” System of Flat Solar Collectors”* ISSN 0003-701X, Applied Solar Energy, 2008, Vol. 44, No. 3, pp. 181–184. © Allerton Press, Inc., 2008.

[9] A. Adhikari, U. R. Dotel, R. Pokhrel, S. T. Hagen *“Thermal Efficiency Test of Locally Manufactured Solar Flat Plate Collector of Nepal”,* 978-1-4673-5556-8/13/ ©2013 IEEE

APPENDIX

Technical Specifications of the compressor:



HIGHLIGHTS	
Type	LBP Type
Voltage(V)	220
Frequency(Hz)	50
COOLING CAPACITY	
kcal/h	92
W	107
Btu/h	365
Power Consumption (W)	96
Refrigerant	R134a
GENERAL SPECIFICATIONS	
Displacement(CC)	4.2
EER Btu/Wh	3.8
Performance (As per ASHRAE)	ASHRAE +7.2°C/54.4°C @ 50 Hz

Table: Technical Specifications of compressor

DS18B20 Temperature Sensor cable specifications:

- Stainless steel tube 6mm diameter by 30mm long
- Cable is 36" long / 91cm, 4mm diameter (1 Meter Long)
- Contains DS18B20 temperature sensor
- Three wires - Red connects to 3-5V, Black connects to ground and White is data.

DS18B20 Sensor Technical specs:

- Usable temperature range: -55 to 125°C (-67°F to +257°F)
- 9 to 12bit selectable resolution
- Uses 1-Wire interface- requires only one digital pin for communication
- Unique 64bit ID burned into chip
- Multiple sensors can share one pin
- ±0.5°C Accuracy from -10°C to +85°C

- Temperature-limit alarm system
- Query time is less than 750ms
- Usable with 3.0V to 5.5V power/data

Specifications of flow rate sensor:

- Model: YF-S201
- Operating Voltage: 5V ~ 24V DC
- Min. Working Voltage: 4.5V DC
- Sensor Type: Hall Effect
- Current Requirement: 15mA @ 5V
- Operating Temperature Range: -25°C ~ 80°C
- Operating Humidity Range: 35% ~ 90% RH
- Water Pressure: ≤ 1.75MPa
- Flow Rate: 1 – 30 L/min
- Cable Length: 15cm
- Connection Thread: 1/2″

Pin Out

RED – Power Input

BLACK – Ground

YELLOW – PWM Output Signal

Technical Specifications of Relay:

- Working voltage: 5V
- Channel: 2 channel
- Item weight: 29g
- Item size: approx. 4.9 * 3.8 * 1.7cm

Features of the Relay:

- High quality double sided compact PCB.
- Opto-Isolated inputs, means your digital electronics is safe and noise free.
- This relay module is 5V active low.
- It is a 2-channel relay interface board, which can be controlled directly by a wide range of microcontrollers such as Arduino, AVR, PIC, ARM, PLC, etc.
- Relay output maximum contact is AC250V 10A and DC30V 10A.
- Standard interface can be directly connected with microcontrollers.

- Red working status indicator lights are conducive to the safe use.
- Widely used for all MCU control, industrial sector, PLC control, smart home control.

Appendix 1- Temperature (Inlet and Outlet of Solar Thermal Collector) and Flow Rate display program:

```
#include <OneWire.h>
// OneWire DS18S20, DS18B20, DS1822 Temperature Example
// http://www.pjrc.com/teensy/td_libs_OneWire.html
// The DallasTemperature library
// http://milesburton.com/Dallas_Temperature_Control_Library

OneWire  ds(8);
volatile int flow_frequency; // Measures flow sensor pulses
unsigned int l_hour; // Calculated litres/hour
unsigned char flowsensor = 2;  // Sensor Input
unsigned long currentTime;
unsigned long cloopTime;
void flow () // Interrupt function
{
   flow_frequency++;
}// on pin 10 (a 4.7K resistor is necessary)

void setup(void) {
  pinMode(flowsensor, INPUT);
   digitalWrite(flowsensor, HIGH); // Optional Internal Pull-Up
   Serial.begin(9600);
   attachInterrupt(0, flow, RISING); // Setup Interrupt
   sei(); // Enable interrupts
   currentTime = millis();
   cloopTime = currentTime;
```

```
}
void loop(void) {
   currentTime = millis();
   // Every second, calculate and print litres/hour
   if(currentTime >= (cloopTime + 1000))
   {
      cloopTime = currentTime; // Updates cloopTime
      // Pulse frequency (Hz) = 7.5Q, Q is flow rate in L/min.
      l_hour = (flow_frequency * 60 / 7.5); // (Pulse frequency x 60 min) / 7.5Q =
flowrate in L/hour
      flow_frequency = 0; // Reset Counter
       }
  byte i;
  byte present = 0;
  byte type_s;
  byte data[12];
  byte addr[8];
  float celsius, fahrenheit;

  if ( !ds.search(addr)) {
   ds.reset_search();
    return;
  }
  for( i = 0; i < 8; i++) {
    Serial.write(' ');
   }

  if (OneWire::crc8(addr, 7) != addr[7]) {
   return;
  }
```

```
// the first ROM byte indicates which chip
switch (addr[0]) {
  case 0x10:

    type_s = 1;
    break;
  case 0x28:

    type_s = 0;
    break;
  case 0x22:

    type_s = 0;
    break;
  default:
    return;
}
ds.reset();
ds.select(addr);
ds.write(0x44, 1);        // start conversion, with parasite power on at the end

delay(100);

present = ds.reset();
ds.select(addr);
ds.write(0xBE);


for ( i = 0; i < 9; i++) {           // we need 9 bytes
```

```
    data[i] = ds.read();
    }
  // Convert the data to actual temperature
  // because the result is a 16 bit signed integer, it should
  // be stored to an "int16_t" type, which is always 16 bits
  // even when compiled on a 32 bit processor.
  int16_t raw = (data[1] << 8) | data[0];
  if (type_s) {
    raw = raw << 3; // 9 bit resolution default
    if (data[7] == 0x10) {
      // "count remain" gives full 12 bit resolution
      raw = (raw & 0xFFF0) + 12 - data[6];
    }
  } else {
    byte cfg = (data[4] & 0x60);
    // at lower res, the low bits are undefined, zero them
    if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution, 93.75 ms
    else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
    else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
    //// default is 12 bit resolution, 750 ms conversion time
  }
  celsius = (float)raw / 16.0;
  fahrenheit = celsius * 1.8 + 32.0;

  Serial.print(celsius);
  Serial.print(" C, ");
  Serial.print(l_hour, DEC); // Print litres/hour
      Serial.println(" L/hour");
      delay(1000);
}
```

Appendix 2- Control System

```
#include <OneWire.h>
OneWire  ds(10);

void setup(void) {
  Serial.begin(9600);
     pinMode(13, OUTPUT);
}

void loop(void) {
  byte i;
  byte present = 0;
  byte type_s;
  byte data[12];
  byte addr[8];
  float celsius, fahrenheit;

  if ( !ds.search(addr)) {
    Serial.println("No more addresses.");
    Serial.println();
    ds.reset_search();
    delay(250);
    return;
  }

  Serial.print("ROM =");
  for( i = 0; i < 8; i++) {
    Serial.write(' ');
    Serial.print(addr[i], HEX);
  }

  if (OneWire::crc8(addr, 7) != addr[7]) {
      Serial.println("CRC is not valid!");
      return;
  }

  switch (addr[0]) {
    case 0x10:
      ;  // or old DS1820
      type_s = 1;
      break;
  case 0x28:

    type_s = 0;
    break;
  case 0x22:

    type_s = 0;
    break;
  default:

    return;
}

ds.reset();
ds.select(addr);
ds.write(0x44, 1);

delay(100);
present = ds.reset();
ds.select(addr);
ds.write(0xBE);


for ( i = 0; i < 9; i++) {           // we need 9 bytes
  data[i] = ds.read();

}

int16_t raw = (data[1] << 8) | data[0];
if (type_s) {
  raw = raw << 3; // 9 bit resolution default
  if (data[7] == 0x10) {
    // "count remain" gives full 12 bit resolution
    raw = (raw & 0xFFF0) + 12 - data[6];
  }
} else {
  byte cfg = (data[4] & 0x60);
  // at lower res, the low bits are undefined, so let's zero them
  if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution, 93.75 ms
  else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
  else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
  //// default is 12 bit resolution, 750 ms conversion time
}
```

```
  celsius = (float)raw / 16.0;
  fahrenheit = celsius * 1.8 + 32.0;
  Serial.print("  Temperature = ");
  Serial.print(celsius);
  Serial.print(" Celsius, ");
                          if (celsius<25)
 {
  digitalWrite(13,LOW);
  delay(100);
}
 else
 if (celsius>36)
 {
 digitalWrite(13,HIGH);
 delay(100);
 }

}
```