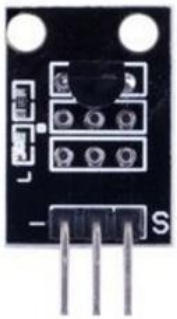
KUMAN
www.kumantech.com

Digital Temperature Sensor

(aka DS18B20)



Overview

The DS18B20 is an accurate, flexible, and inexpensive digital thermometer. Its wear-resistance and impact-resistance make it suitable for portable or encapsulated applications in high-stress temperature management environments, such as in cable trenches, blast furnaces, boilers, machine rooms, greenhouses, clean rooms, ammunition stores, and so on. In this experiment, you will have the Raspberry Pi report the current temperature (in Celcius) measured by the digital temperature sensor.

Experimental Materials

```
Raspberry Pi            x1
Breadboard              x1
DS18B20 module          x1
Dupont jumper wires
```

Experimental Procedure

1. If you have not done so already, prepare your development system by installing the Python interpreter, RPi.GPIO library, and wiringPi library as described in READ_ME_FIRST.TXT.

2. Install the DS18B20 digital temperature sensor on your breadboard, and use Dupont jumper wires to connect it to your Raspberry Pi as illustrated in the Wiring Diagram below.

3. Execute the sample code stored in this experiment's subfolder.
   If using C, compile and execute the C code:

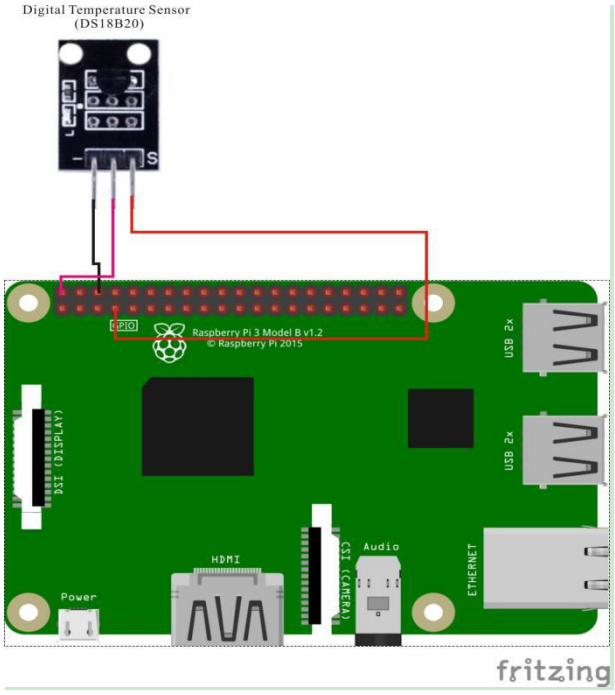
```
cd Code/C
gcc ds18b20.c -o ds18b20.out -lwiringPi
./ds18b20.out
```

   If using Python, launch the Python script:

```
cd Code/Python
python ds18b20.py
```

4. Make experimental observations. The temperature is displayed on the Raspberry Pi command line interface.

Wiring Diagram



DS18B20 pin position:

"S" ↔ Raspberry Pi pin 7 (see note below)

"+" ↔ Raspberry Pi +5V

"-" ↔ Raspberry Pi GND

Technical Background

The DS18B20 uses a single wire for bidirectional communication with the Raspberry Pi or other controller (in what is called a “single-pin bus interface”). It can be configured to report in differing amounts of temperature resolution, and can be networked with other DS18B20s in highly-parallel thermometry applications. It has a wide temperature range with ±0.5°C accuracy in the -10°C to +85°C range, and can be powered through its own data line while preserving its internal state and configuration parameters across power outage. For more information on using this flexible thermometer, search the internet for “DS18B20 datasheet”.
Note that the Raspberry Pi’s code libraries *only* work with the DS18B20 “S” pin connect to GPIO7. If you design your own experiments for DS18B20 or write your own DS18B20 source code, do not change this pin mapping.

Sample Code

Python Code

```
   #!/usr/bin/env python

   # Note:
   # ds18b20's data pin must be connected to pin7.
   # Reads temperature from sensor and prints to stdout
   # id is the id of the sensor

   import os
   import time

   def readSensor(id):
      tfile = open("/sys/bus/w1/devices/"+id+"/w1_slave")
      text = tfile.read()
      tfile.close()
      secondline = text.split("\n")[1]
      temperaturedata = secondline.split(" ")[9]
      temperature = float(temperaturedata[2:])
      temperature = temperature / 1000
      print "Sensor: " + id  + " - Current temperature : %0.3f
C" % temperature


   # Reads temperature from all sensors found in
/sys/bus/w1/devices/
   # starting with "28-...
```

```
def readSensors():
    count = 0
    sensor = ""
    for file in os.listdir("/sys/bus/w1/devices/"):
        if (file.startswith("28-")):
            readSensor(file)
            count+=1
    if (count == 0):
        print "No sensor found! Check connection"

# read temperature every second for all connected sensors
def loop():
    while True:
        readSensors()
        time.sleep(1)

# Nothing to cleanup
def destroy():
    pass

# Main starts here
if __name__ == "__main__":
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

C Code

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>

#define  BUFSIZE  128

char Buff[256] = {0};
```

```
char* get_ds18b20_id(char *DirName)
{
   DIR *dirp = NULL;
   struct dirent *Filep = NULL;

   dirp = opendir(DirName);
   while(1)
   {
      Filep = readdir(dirp);
      strcpy(Buff, Filep->d_name);
      if(!strncmp(Buff, "28-", 3))
      {
         return Buff;
      }
   }

}

int main(void)
{
   float temp;
   int i, j, fd, ret;

   char buf[BUFSIZE];
   char tempBuf[5];
   char fileName[256] = {0};
   char Dir[256] = "/sys/bus/w1/devices/";
   char *ds18b20_id = NULL;

   ds18b20_id = get_ds18b20_id(Dir);
   strcpy(fileName, Dir);
   strcat(fileName, ds18b20_id);
   strcat(fileName, "/w1_slave");
   printf("filename is %s\n", fileName);
   while(1)
   {
      if((fd = open(fileName, O_RDONLY)) == -1)
      {
         perror("open device file error");
         return -1;
      }
```

```
        while(1)
        {
           ret = read(fd, buf, BUFSIZE);
           if(0 == ret)
           {
              break;
           }
           if(-1 == ret)
           {
              if(errno == EINTR)
              {
                 continue;
              }
              perror("read()");
              close(fd);
              return -1;
           }
        }
        for(i=0;i<sizeof(buf);i++)
        {
           if(buf[i] == 't')
           {
              for(j=0;j<sizeof(tempBuf);j++)
              {
                 tempBuf[j] = buf[i+2+j];
              }
           }
        }

        temp = (float)atoi(tempBuf) / 1000;

        printf("%.3f C\n",temp);

        sleep(1);

        close(fd);
    }

    return 0;
}
```