Arkal Filter Control

Contact Energy

Filter Control Board

https://github.com/optimho/Steamfield-Arkal-Filter-Pack-Control

Table of Contents

Table of Contents

[Operation 1](#_Toc416092940)

[Control 1](#_Toc416092941)

[Settings 1](#_Toc416092943)

[Connection Diagram 1](#_Toc416092944)

[Maintenance 2](#_Toc416092945)

[Schematics 3](#_Toc416092946)

[Firmware Listing 4](#_Toc416092947)

Operation

t

He Filter control module monitors the differential pressure across the filter Pack and also counts the amount of filter flush cycles. If the filter pack is constantly flushing and not able to clear the high differential, then the bypass will be opened and an alarm will be raised.

Refer to the Arkal filter documentation to maintain a blockage in the filter.

All documentation for the project is available on Github, follow the following URL

<https://github.com/optimho/Steamfield-Arkal-Filter-Pack-Control>

Control

The reset button will be illuminated on the front of the panel if the filter control has operated the bypass valve. To resume normal operation reset the filter control by pressing the illuminated button. The light should extinguish and the bypass valve should close, if the DP (Differential Pressure) has returned to normal.

Settings

The board was originally designed with dipswitches, but the dipswitch was resigned and the firmware was added to the project. All changes and customization can be done directly in the firmware.

If you want to modify or monitor the controller install the Simple IDE software that is available at the Github address above under the firmware folder.

# Connection Diagram

🖎

Maintenance

There is no maintenance required apart from testing and inspecting the operation of the controller at regular intervals.

Schematics

Firmware Listing

/\*

\*

\*SteamFileldController.c

\*

\*Main listing

Steamfield Controller Firmware

controls the bypass valve, opens if the flush cycles are to numerous.

functions recordEvent( period) this function will return the amount of operations preformed in a specified period in seconds

The function is a low speed function and cannot really handle inputs at a frequency less than 1 second.

\*/

#include <TimeFile.h>

#define period 90 // This is the period that the amount of flush starts are counted

#define diffAlarm 10 // This is the period that initiates a diff Alarm

#define flushValve 26 //

#define highDiff 1 //

#define filter1 2 //define the micro processor pin usage

#define filter2 3 //

#define filter3 4 //

#define clear 5 // Reset input

#define FilterStarts 5

char \*timeOfEvent;

int i=0;

int placeHolder1=0;

int placeHolder2=0;

int placeHolder3=0;

int diffState=0; // Time differential pressure became high

int diffTime=0;

int currentTime=0; // The amount of seconds since the last time reset or switch on

int amountOfEvents=0; // amount of flush operations recorded in a specified period

int main()

{

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

print("\*\*\*\*\*\*\* Steamfield \*\*\*\*\*\n");

print("\*\*\*\* Arkal filter controller \*\*\*\n");

print("\*\*\*\* Contact Energy \*\*\*\* \n");

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

print("\*\*\*\*\*\* Michael \*\*\* \n");

print("\*\*\*\*\*\* Rick \*\*\* \n");

print("\*\*\*\*\*\* Grant \*\*\* \n");

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

print("\*\*\*\* 15-03-2015 version 1.0 \*\* \n");

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

print(" \n");

print(" \n");

pause(1000);

for(;;)

{

int diff = input(highDiff);

int flush\_1 = input(filter1);

int flush\_1\_old;

int flush\_2 = input(filter2);

int flush\_2\_old;

int flush\_3 = input(filter3);

int flush\_3\_old;

int clr = input(clear);

currentTime=timeNow(); // the time now is currentTime

/\* This section is to determine if a flush cycle has been completed.

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if (placeHolder2 != sequenceCheck(flush\_1, flush\_2, flush\_3))

{

//print(" Flush sequence complete %d \n", sequenceCheck(flush\_1, flush\_2, flush\_3));

}

placeHolder2=sequenceCheck(flush\_1, flush\_2, flush\_3);

/\* Prints to terminal some program information

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if (placeHolder1 != indexer())

{

print("number of filter starts %d per %d seconds - filter %d %d %d \n", indexer(), period, flush\_1, flush\_2 ,flush\_3);

}

placeHolder1=indexer();

//printf("amount of triggers per period %i \n", amountOfEvents);

// printf("Seconds since last reset %i \n",currentTime);

// print("diff Pressure = %d\n", diff);

// print("flush 1 = %d\n", flush\_1);

// print("flush 2 = %d\n", flush\_2);

// print("flush 3 = %d\n", flush\_3);

// print("Diff High since = %d \n", diffState);

// print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

//print(" \n");

/\* This section is to determine if a flush has been initiated.

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if ((flush\_1 > flush\_1\_old) || (flush\_2 > flush\_2\_old) || (flush\_3 > flush\_3\_old))

{

amountOfEvents=recordEvent();

}

flush\_1\_old=flush\_1;

flush\_2\_old=flush\_2;

flush\_3\_old=flush\_3;

/\* This section is to determine how long the high diff alarm has been active.

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if ((diff==1) && (diffState==0)) //If there is a high diff and there is not currently a high diff already then save the time

{

diffState = currentTime;

//diffTime = diffState; // That the diff went high

}

else if ((diff==1) && (diffState>0)) //If there is a high diff and there is not currently a high diff already then save the time

{

diffTime = currentTime-diffState; // That the diff went high

}

else if ((diff==0)&& (diffState>0) ) // If the diff clears and the diff was high consider that the high diff to have cleared.

{

diffTime = 0;

diffState = 0;

//

}

/\* This program segment alarms if the filter starts per time period is exceeded. \*/

if (placeHolder1>FilterStarts)

{

high(flushValve);

}

/\* This code segment is the Reset part that resets all counters

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if (clr==1)

{

clrEvents();

low(flushValve);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Print to terminal if DIFF pressure is high.

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if ((diffTime > diffAlarm))

{

print("diff Alarm %d \n",currentTime );

pause(1000);

}

placeHolder3=diff;

/\* This section of the code is to zero the clock periodially so that you do not get any math mathmatical errors if the clock rolls over to zero.

/\* This decison to zero the clock is only made when all variables are in the zero state past some time in the future .

\* Diff pressure is reset and there has been no flush cycles in the last period and the time is > than 1000000 seconds

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if ((diff==0) && (diffState==0) && (currentTime>1000000) && (amountOfEvents==0))

{

settime();

}

/\* This code must be called regularly to make sure that only the events within the specified period are recorded -- this filter also put

\* a reasonable processor overhead so it is good to call it once per main loop period so that each loop iterates at the same speed.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

reducer(period); //in a period

}

return 0;

}

-----------------------------------------------------------

\*/

TimeFile.c

\*/

#include <TimeFile.h>

#define BUFFERLEN 10

//time\_t now;

int timeNow(void)

{

// time\_t now;

now = time(NULL);

// printf("%s", asctime(localtime(&now)));

// printf("%i \n", now);

return now;

}

void printtime(void)

{

now = time(NULL);

printf("%i", &now);

return asctime(localtime(&now));

}

int getdst(void)

{

int rc = 0;

printf("Use daylight savings time [y/n] ?");

fflush(stdout);

rc = (getchar() == 'y') ? 1 : 0;

getchar();

return rc;

}

int prompttime(char \*prompt)

{

int rc = 0;

char \*endp;

char buffer[BUFFERLEN];

do {

printf("Enter %s: ",prompt);

fflush(stdout);

fgets(buffer,BUFFERLEN,stdin);

fflush(stdin);

rc = strtol(buffer, &endp, 10);

if(endp == buffer)

{

if('\n' == \*endp)

{

rc = 0;

break;

}

printf("Invalid entry \"%c....\" Please enter a number.\n", \*endp);

}

} while(endp == buffer);

return rc;

}

void settime(void) // set time virtual RTC

{

struct timeval tv;

struct tm t;

t.tm\_isdst = 0; //getdst();

t.tm\_year = 1968-1900; //prompttime("Year")-1900;

t.tm\_mon = 11-1; //prompttime("Month")-1;

t.tm\_mday = 11; //prompttime("Day of the month");

t.tm\_hour = 12; //prompttime("Hour");

t.tm\_min = 0; //prompttime("Minute");

t.tm\_sec = 0; //prompttime("Second");

tv.tv\_sec = mktime(&t);

settimeofday(&tv, 0);

// printf("Set time all done.\n");

}

int recordEvent()

{

int a=indexer();

currentEventTime[a]=timeNow();

return indexer();

}

/\* How many items in the array, count any non zero entries\*/

int indexer()

{

int i=0;

while (currentEventTime[i]!=0)

{

i++;

}

return i;

}

/\*Filter through the EventTime array and remove events that are older than a value in seconds \*/

void reducer(period)

{

int interval=period;

int currentTime = timeNow();

int Events=100;

int i=0;

while (currentEventTime[i]!=0)

{

i++;

}

i=0;

if ((currentTime-interval>currentEventTime[0]) && (currentEventTime[0]>0))

{

shiftArray(indexer());

}

}

shiftArray(int numberOfElements)

{

for (int i=1; i<=numberOfElements; i++)

{

currentEventTime[i-1]=currentEventTime[i];

}

}

int sequenceCheck(flush1, flush2, flush3)

{

if ((flush1==1) && (gv\_sequence==0))

{

gv\_sequence=1;

return gv\_sequence;

}

else if ((flush2==1) && (gv\_sequence==1))

{

gv\_sequence=2;

return gv\_sequence;

}

else if ((flush3==1) && (gv\_sequence==2))

{

gv\_sequence=3;

return gv\_sequence;

}

else

{

gv\_sequence=0;

return gv\_sequence;

}

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*void clear records \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void clrEvents()

{

int i=0;

while (currentEventTime[i]!=0)

{

currentEventTime[i]=0;

i++;

}

}

/\*

\* Header File

\*

\* TimeFile.h

\*

\*

\*/

#include <time.h>

#include <sys/time.h>

#include <simpletools.h>

#define TRUE 1

#define FALSE 0

#define FAIL 0

time\_t now;

int gv\_sequence; // Global variable for the sequence function

volatile int currentEventTime[1000];

int \*addr;

int timeNow(void); /\* The time now is the time since the epoch which is 1 January 00:00:00 1970 \*/

void shiftArray(int numberOfElements); /\* Rapidly shift all the elements of an array by one/\*/

volatile struct timeStruct{

int hour;

int min;

int sec;

}EventTime[10];

void printtime(void);

int getdst(void);

int prompttime(char \*prompt);

void settime(void);

void starts\_per\_hour( int i);

int recordEvent(); /\* inserts a time event onto an array and returns an integer of the amount of events per

period in seconds \*/

int indexer(); /\* counts how many entries are in the array \*/

void reducer(int period); /\* keeps all entries current within the last period period is in seconds \*/

int totSec(int i); /\* calculates the total amount of seconds in EventTime[x] \*/

int sequenceCheck(int flush1,int flush2,int flush3); /\* Returns True if a flush cycle is complete, that means that

Filter1 then filter 2 then after that filter 3 \*/

void clrEvents(); /\* Clears all records back to zero .\*/