Исследование возможности создания датчика горизонта для системы ориентации малого космического аппарата на основе инфракрасного датчика температуры

Федоров Вячеслав Васильевич

Физический факультет. Оптический практикум. 4 семестр.

Группа 16362, 2018 г.

Научный руководитель:

А. М. Задорожный, к.ф.-м.н., доцент, зав. ОАИ НГУ

Аннотация

В данной курсовой работе было проведено исследование возможности создания датчика горизонта Земли на основе инфракрасной (ИК) матрицы MLX90621 с разрешением 16×4 пикселей и микроконтроллером ATmega328 на платформе Nano без подвижных элементов и с легкой, малогабаритной конструкцией. ИК матрица была согласована с микроконтроллером с помощью написанного и отлаженного программного обеспечения на языках программирования Arduino и C++. Данная система была протестирована в лабораторных условиях. Результаты эксперимента подтвердили правильную работу системы. Настоящая курсовая работа является *базовым этапом* в конструировании датчика горизонта для системы ориентации малого космического аппарата.

Ключевые слова: инфракрасный датчик температуры, малый космический аппарат, наноспутник, датчик горизонта, система ориентации.

Работа выполнена в отделе атмосферных исследований НГУ.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Физический факультет
Кафедра общей физики

Федоров Вячеслав Васильевич

КУРСОВАЯ РАБОТА

«Исследование возможности создания датчика горизонта для системы ориентации малого космического аппарата на основе инфракрасного датчика температуры»

2 курс, группа 16362

Научный руководитель:
к.ф.-м.н., доцент, зав. ОАИ НГУ
_____ А. М. Задорожный
«____»_____ 2018 г.
Оценка научного руководителя
_____

Преподаватель практикума:
д.ф.-м.н., доцент, в.н.с. ИЯФ СО РАН
_____ Н. Ю. Мучной
«____»_____ 2018 г.
Оценка преподавателя практикума
_____

Новосибирск, 2018 г.

Аннотация

В данной курсовой работе было проведено исследование возможности создания датчика горизонта Земли на основе инфракрасной (ИК) матрицы MLX90621 с разрешением 16×4 пикселей и микроконтроллером ATmega328 на платформе Nano без подвижных элементов и с легкой, малогабаритной конструкцией. ИК матрица была согласована с микроконтроллером с помощью написанного и отлаженного программного обеспечения на языках программирования Arduino и C++. Данная система была протестирована в лабораторных условиях. Результаты эксперимента подтвердили правильную работу системы. Настоящая курсовая работа является *базовым этапом* в конструировании датчика горизонта для системы ориентации малого космического аппарата.

Ключевые слова: инфракрасный датчик температуры, малый космический аппарат, наноспутник, датчик горизонта, система ориентации.

Работа выполнена в отделе атмосферных исследований НГУ.

Оглавление

1. Введение

Инфракрасное излучение (ИК) – это электромагнитное излучение, которое находится в спектре между красным концом видимого спектра, длина волны $\lambda \approx 0{,}7$ мкм, и микроволновым излучением, с длиной волны $\lambda \approx 1$ мм. ИК излучение является синонимом теплового излучения, так как человек воспринимает инфракрасное излучение, которое исходит от нагретых тел, как тепло. Причем, чем горячее тело, тем меньше длина волны ИК излучения.

Человек не способен воспринимать ИК излучение в отличии, например, от комаров, рыб или змей. Но с помощью ИК матриц и последующей специальной обработкой, перевод полученных данных с матриц в цифры, значения температуры, или в цвет, инфракрасное излучение становится доступным человеку.

Тепловая или ИК съемка с космического аппарата основана на фиксации инфракрасного излучения Земли, вызванного солнечной радиацией или эндогенным теплом.

Инфракрасную видеонавигацию для космических аппаратов (КА) и малых космических аппаратов (МКА) используют с конца 20 века [1]. Признание получили оптические построители местной вертикали (ПВМ). Их работа основана на контрасте температур между поверхностью Земли и «холодным» космосом, благодаря чему возможно ведение видимого горизонта Земли в 3-х и более различных точках. Температура поверхности Земли около 300 К, температура космического пространства около 3 К.

Существуют разные варианты реализации ИК визирования горизонта. Самым распространенным является система кругового сканирования горизонта планеты ИК камерой. Оптическая ось данного построителя вращается, описывая конус, ось которого совпадает с осью космического аппарата. Так обеспечивается круговой просмотр края горизонта Земли. Но системы такого типа обладают существенными недостатками, такими как: высокое потребление энергии, создаваемый момент, который вращает КА, а также громоздкость установки. Приведенные недостатки говорят о нерациональности использования данного

ПВМ на МКА, где сильно ограничены энергетика и габариты.

Для МКА идеально подходят неподвижные малые ПВМ с низким энергопотреблением. Здесь существуют два принципиальных подхода. В первом варианте ИК сенсоры (от 4-х до 9 и более штук) располагаются на одной стороне МКА, но под разными углами и с разными углами раствора [2]. Схема такого датчика направления на Землю (в надир) приведена на рис. 1. В разные ИК сенсоры попадет либо поверхность Земли, либо космическое пространство, либо поверхность Земли и космическое пространство. После обработки полученной информации со всех сенсоров однозначно определяется направление на Земли.

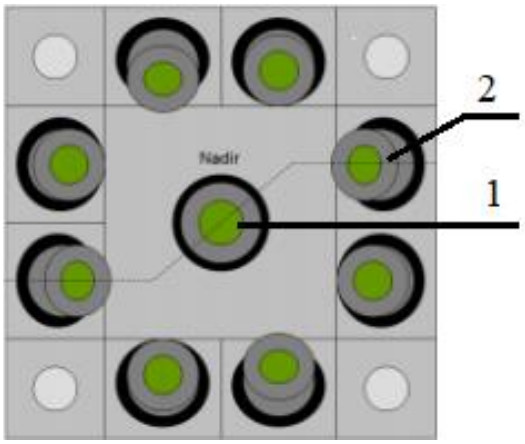

Рис. 1. Схема датчика направления на Землю 1: 1 – центральный ИК сенсор, 2 – перифирийный ИК сенсор.

Во втором варианте два ИК матричных датчика температуры, являющихся по сути датчиками горизонта, находятся на разных гранях МКА так, что их оптические оси (оси визирования) перпендикулярны [3]. На изображениях, полученных с ИК матриц, определяют линии горизонта и с помощью формул аналитической геометрии находят направление в надир. Схема датчика направления в надир такого типа приведена на рис. 2.

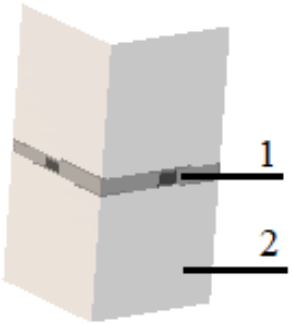

Рис. 2. Схема датчика направления на Землю 2: 1 – ИК матричный сенсор, 2 – корпус МКА.

В данной работе будет рассматриваться второй вариант расположения ИК сенсоров из-за его простоты и надежности.

Таким образом, *цель работы* состоит в исследование возможности создания датчика горизонта для МКА с помощью легкой оптической системы без

подвижных частей и малым энергопотреблением, а также ее практическая реализация.

В статье «Определение направления на местную вертикаль для наноспутника класса CubeSat по анализу изображения Земли» [3] рассмотрена система определения направления на Землю, подобная приведенной на рис. 2. В статье представлены основные формулы и результаты численного моделирования для угла полураствора камеры 28 градусов и высоты спутника 300 км. Проведенное в статье исследование показало, что данный алгоритм имеет *широкую область применения* и его *можно использовать* для определения ориентации МКА. Результаты численного моделирования для одной камеры приведены на рис. 3, где цветом показан угловой размер дуги горизонта, попадающей в кадр камеры; тангаж и рыскание задают направление оси визирования камеры. Красный цвет

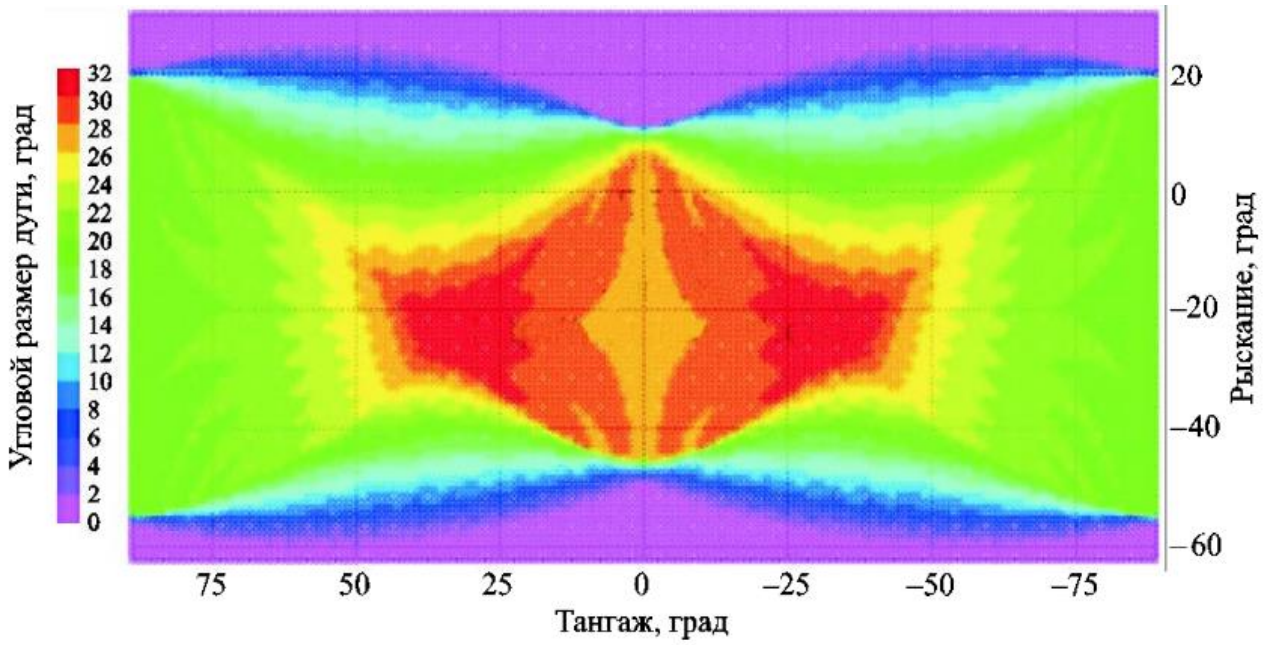

Рис. 3. Результаты численного моделирования для области применимости СО.

на рис. 3 соответствует прохождению линии горизонта Земли по диагонали кадра. Для второй камеры результат будет аналогичным, только повернутым на 90 градусов по часовой стрелке относительно начала координат. Итоговым результатом для двух камер будет являться пересечение областей применимости двух камер.

2. Разработка ИК датчика для системы ориентации МКА

Сначала обоснуем выбор ИК матрицы и микроконтроллера, используемых в работе; затем приведем схемы подключения и программу, написанную на C++ для датчика, а также основные погрешности ИК датчика.

2.1. Выбор ИК матрицы

Для ИК датчика была выбрана инфракрасная матрица MLX90621. Технические характеристики представлены в таблице 1. Это откалиброванная ИК матрица с разрешением 16×4 пикселей в свинцовом корпусе TO-39 с низким уровнем шума и прецизионным АЦП. Она содержит 2 микросхемы в одном корпусе: MLX90670 (ИК матрица с электроникой формирования сигнала) и чип памяти 24AA02 (256x8 EEPROM). Для измерения температуры окружающей среды чипа в него встроен датчик температуры PTAT (Proportional To Absolute Temperature). Выходы датчиков ИК и PTAT хранятся во внутренней памяти и доступны через I2C

Таблица 1. Технические характеристики MLX90621

Параметр	Значение
Калиброванный диапазон измерения температур	От -70 до +300 °C
Точность измерения температур	0,5 °C
Диапазон рабочих температур	От -40 до +85 °C
Поле зрения	60° горизонт, 15° по вертикали
Разрешение	16×4 точек
Напряжение питания, VDD	2,6 В
Ток питания, IDD (без нагрузки)	5-9 мА
Масса	≤ 2 гр.

Окончательная обработка данных с ИК матрицы осуществляется внешним микроконтроллером, который вычисляет температуру каждого элемента матрицы считывая исходные данные из ОЗУ и учитывая данные калибровки, хранящихся в памяти EEPROM, доступные через шину I2C.

Преимущества данной матрицы очевидны – это малый размер и низкая стоимость, легкая интеграция в схемы, заводская калибровка ИК измерения

температуры и промышленный стандарт TO-39.

2.2. Выбор микроконтроллера

В качестве микроконтроллера выбран ATmega328 в составе платы Arduino Nano. Плата Arduino Nano имеет небольшие размеры и ее характеристик достаточно для поставленных задач. Arduino Nano – это одноплатный контроллер с открытыми начальными кодами. Платформа Nano, построенная на микроконтроллере ATmega328, может использоваться в лабораторных работах.

Краткие технические характеристики Arduino Nano приведены в таблице 2. ATmega328 поддерживает интерфейсы I2C. Также в Arduino включена библиотека Wire для удобства использования шины I2C. Библиотека Wire будет активно использоваться при написании программы для использования ИК матрицы. Платформа программируется посредством специального программного обеспечения (ПО) Arduino, а выбранный микроконтроллер имеет записанный загрузчик, который облегчает запись новых программ без использования сторонних программаторов. Arduino Nano питается через интерфейс Mini-B USB.

Таблица 2. Технические характеристики Arduino Nano

Параметр	Значение
Микроконтроллер	ATmega328
Рабочее напряжение	5 В
Постоянный ток через вход/выход	40 мА
Входное напряжение (рекомендуемое)	7 – 12 В
Входное напряжение (предельное)	6 – 20 В
Флеш-память	32 Кб
ОЗУ	2 Кб
EEPROM	1 Кб
Тактовая частота	16 МГц
Размеры	1.85×4.2 см

Для упрощения монтажа и отладки схемы включения MLX90621 используется беспаечная макетная плата. Это пластиковое основание со множеством отверстий с металлическими клипсами внутри. Благодаря ей можно совсем

отказаться от пайки, что упрощает монтаж схемы и позволяет проводить отладочные работы большое количество раз. Внешний вид матрицы MLX90621 и платы Arduino Nano показан на рис. 4.

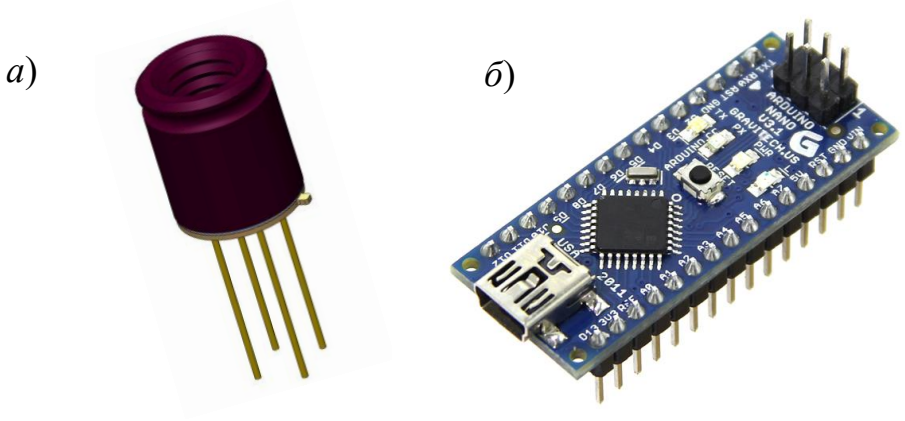

Рис. 4. Внешний вид: *а*) MLX90621; *б*) Arduino Nano.

2.3. Лабораторный стенд для исследования ИК матрицы

ИК матрица MLX90621 подключается к шине I2C, следовательно, её легко подключить к Arduino Nano. Схема подключения матрицы показана на рис. 5.

MLX90621 питается от напряжения 2,6 В, а микроконтроллер выдает 3,3 В. Самый простой способ получить нужное напряжение – подключить ИК матрицу к порту Arduino 3,3 В через кремниевый диод. На кремниевом диоде как известно

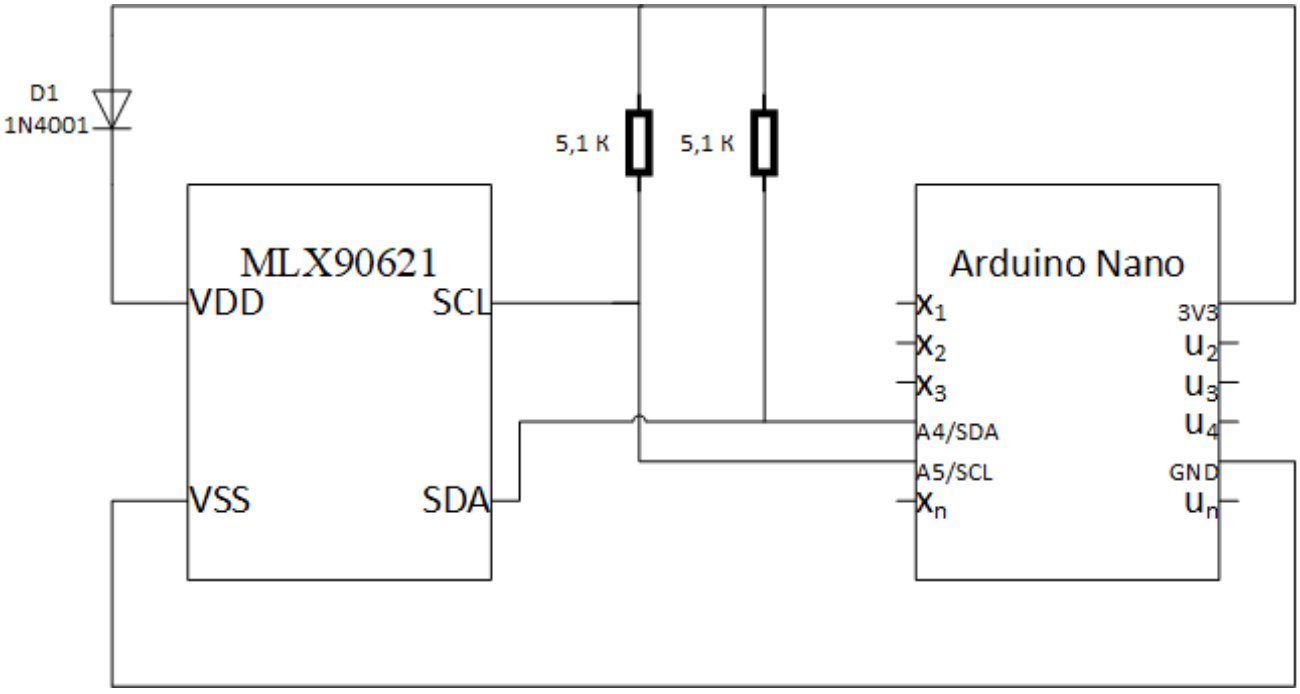

Рис. 5. Принципиальная схема подключения MLX90621 к Arduino Nano.

падает около 0,6 В. На ножке питания матрицы будет около 2,7 В. Также понадобятся два подтягивающих резистора с номиналом 5,1 К. Это нужно для поддержания высокого логического уровня в случае разрыва контакта с логическим выходом ИК матрицы. Ножки MLX90621 SLC и SDA подключаем к входам A4 и A5 Arduino Nano соответственно и подтягиваем их к шине питания сопротивлениями 5,1 К.

ИК матрица определяется как два устройства с адресами 0x50 и 0x60. По первому адресу находится ПЗУ с калибровочными константами для каждого из 64-х пикселей матрицы, а также для встроенного датчика внешней температуры. По второму адресу производится непосредственное чтение температуры. Для перевода полученного массива чисел в массив температур понадобятся формулы из спецификации MLX90621 [4]. Для управления матрицей и обработки исходных данных был написан скетч (программа) на внутреннем языке платформы Arduino и библиотека на языке программирования C++. Главная задача написанного ПО заключается в калибровке исходных данных MLX90621, т.е. расчёт правильных, действительных значений температур. Коды программ и краткие комментарии к нему приведены в приложении к курсовой работе. На выходе в терминале Arduino выводится массив температур 16×4.

К настоящему времени написано и полностью отлажено ПО для управления ИК матрицей; а также проведено первичное тестирование матрицы. Пример теста приведен на рис. 6 и 7. Здесь для теста MLX90621 проведен простой эксперимент. ИК матрица была направлена на голову экспериментатора с расстояния вытянутой руки. Зная размер головы и расстояние до матрицы, соответственно 20 и 60 см, путем простых вычислений не трудно узнать угловой размер головы. Он составил около 0,3 радиан. Раствор камеры примерно 1 радиан, количество пикселей по горизонтали 16, а по вертикали 4, следовательно, изображение будет занимать 5 пикселей по горизонтали кадра и занимать весь кадр по вертикали. Температура открытых участков человеческого тела около 36 °C, а температура в лаборатории примерно 27 – 30 °C.

Результат эксперимента приведен на рис. 6 в матричном виде и на рис. 7 в

графическом исполнении. Графика была быстро получена средствами Microsoft Word. Хорошо видно, что по центру кадра находится объект с

```
[31.25,31.68,29.32,30.51,29.66,29.25,32.78,35.36,35.57,35.58,33.83,28.83,27.55,30.11,29.13,30.57]
[29.53,30.08,28.14,28.26,28.78,29.86,33.57,36.69,35.37,36.09,36.17,29.68,28.68,28.91,28.65,29.66]
[29.80,29.01,30.02,29.25,28.55,30.16,36.11,36.45,35.63,34.78,34.63,28.24,28.57,29.68,30.52,29.09]
[31.29,30.53,29.20,29.69,29.09,29.54,34.52,36.68,36.02,35.25,31.15,28.65,29.01,30.01,30.00,30.15]
```

Рис. 6. Человек в поле зрения ИК матрицы в матричном виде.

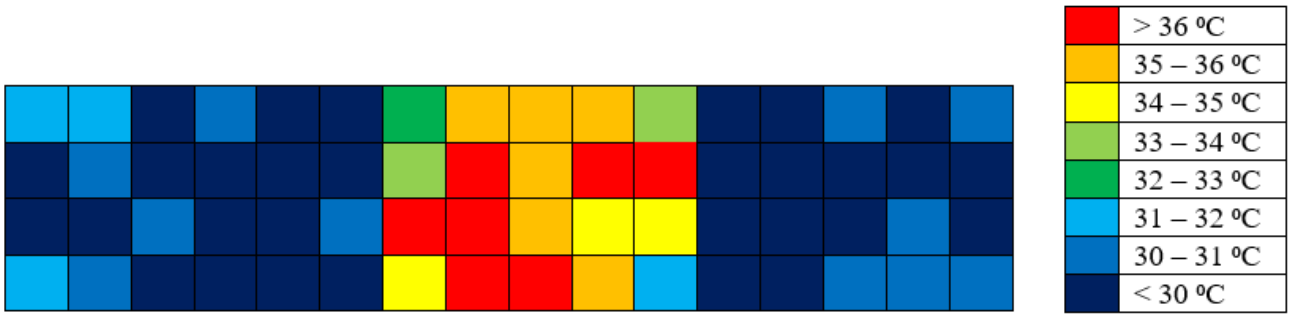

Рис. 7. Человек в поле зрения ИК матрицы в графическом виде.

температурой около 36 °C, который занимает 5 пикселей по горизонтали кадра и все поле кадра по вертикали, на фоне со средней температурой примерно 30 °C.

3. Обсуждение полученных результатов

Заявленная точность сенсора составляет 0,5 °K при частоте обновления 1 Гц [4]. Этого вполне достаточно для поставленной задачи выделить «горячую» Землю со средней температурой 300 °K на фоне «холодного» космического пространства с температурой 3 °K. Однако, стоит иметь в виду, что данной точности не хватает для применения MLX90621 в медицине, что и написано в спецификации к данной матрице. Также нужно отметить, что температура поверхности открытых участков человеческого тела может сильно отличаться от 36,6 °C.

Результаты поставленного эксперимента (рис. 7) качественно подтвердили возможности данной системы. А именно, возможность отделять более нагретые тела от менее нагретых тел. В эксперименте разница температур составила всего 6 °C, отчетливо видны перепады температур в 1 °C, что является отличным результатом с точки зрения планируемого использования матрицы для регистрации границы областей с температурами, отличающимися примерно на 300 °C. В ближайшее время запланирован эксперимент с жидким азотом,

температура кипения которого составляет −196 °C. Это позволит создать модель «теплой» Земли и провести тестирование ИК матрицы в условиях, наиболее приближенных к реальным условиям эксплуатации данной системы на борту малого космического аппарата.

Стоит отметить, что наблюдать числа в терминале Arduino не удобно и утомительно. Предлагается написать программу на высокоуровневом языке программирования Python 3,5, с помощью которой удалось бы рисовать двумерные изображения, подобные приведенному на рис. 7, в реальном времени на основе получаемых матриц температур из ИК матрицы. Работа уже ведется.

4. Выводы и заключение

В данной курсовой работе было проведено исследование возможности создания датчика горизонта Земли для системы ориентации малого космического аппарата на основе ИК датчика температуры с малым энергопотреблением, без подвижных элементов и с легкой, малогабаритной конструкцией.

Была обосновано выбрана подходящая для поставленных задач ИК матрица (MLX90621) и микроконтроллер ATmega328 на платформе Arduino Nano. Началась практическая разработка датчика горизонта. ИК матрица была согласована с микроконтроллером с помощью написанного и отлаженного ПО. Система была протестирована в лабораторных условиях. Она оказалась способна видеть разницу температур в единицы градусов, что является очень хорошим результатом с точки зрения планируемого использования матрицы для определения горизонта «горячей» Земли (~ 300 °К) на фоне «холодного» космического пространства (~ 3 °К). Результаты эксперимента подтвердили правильную работу системы.

Все использованные элементы, а именно: MLX90621, Arduino Nano, резисторы, диод и др., легкодоступны и достаточно дешевые.

Таким образом, настоящая курсовая работа является *базовым этапом* в конструировании датчика горизонта для МКА.

Данные, полученные в работе, *будут использованы* в ОАИ НГУ при конструировании МКА класса CubeSat. Запуск спутника запланирован на 2020 г.

Список литературы

1. Волков В. Г., Ковалев А. В., Федчишин В. Г. Тепловизионные приборы нового поколения. // Специальная техника, 2001, № 6, С. 16–21.

2. Siegfried W., Janson, Brian S., Hardy, Andrew Y. Attitude Control on the Pico Satellite Solar Cell Testbed-2ю. // 26th Annual AIAA/USU Conference on Small Satellites.

3. Ломака И. А., Устюгов Е. В. Определение направления на местную вертикаль для наноспутника класса CubeSat по анализу изображений Земли. //Инженерный журнал: наука и инновации, 2016, вып. 8.

4. Спецификация к MLX90621// Datasheet_90621.

Приложение

```
//скетч Arduino readTemperatures.ino
//подключение библиотек
#include <SPI.h>
#include <Arduino.h>
#include <Wire.h>
#include "MLX90621.h"

MLX90621 sensor; // создаем класс
void setup(){
  Serial.begin(19200);
  Serial.println("trying to initialize sensor...");
  sensor.initialise (16); // start the thermo cam with 8 frames per second
  Serial.println("sensor initialized!");
 }
void loop(){
  sensor.measure(true); //получаем новые показания с датчика
  for(int y=0; y<4; y++){ //пробегаем все строки
    Serial.print("[");
    for(int x=0; x<16; x++){ //пробегаем все столбцы
      Serial.print(sensor.getTemperature(y+x*4));
      if (x<15)
      Serial.print(",");
    Serial.print("]");
    if (y<3)
    Serial.print("\n");
  }
  Serial.print("\n\n");
 delay(31);
};
```

```
// Заголовочный файл MLX90621.h
#ifndef MLX90621_H_
#define MLX90621_H_
#ifdef __cplusplus

// Библиотеки
#include <Arduino.h>
#include <Wire.h>

// Начало регистров
#define CAL_ACOMMON_L 0xD0
#define CAL_ACOMMON_H 0xD1
#define CAL_ACP_L 0xD3
#define CAL_ACP_H 0xD4
#define CAL_BCP 0xD5
#define CAL_alphaCP_L 0xD6
#define CAL_alphaCP_H 0xD7
#define CAL_TGC 0xD8
#define CAL_AI_SCALE 0xD9
#define CAL_BI_SCALE 0xD9


#define VTH_L 0xDA
#define VTH_H 0xDB
#define KT1_L 0xDC
#define KT1_H 0xDD
#define KT2_L 0xDE
#define KT2_H 0xDF
#define KT_SCALE 0xD2
```

```
// Общие коэффициенты чувствительности
#define CAL_A0_L 0xE0
#define CAL_A0_H 0xE1
#define CAL_A0_SCALE 0xE2
#define CAL_DELTA_A_SCALE 0xE3
#define CAL_EMIS_L 0xE4
#define CAL_EMIS_H 0xE5
#define CAL_KSTA_L 0xE6
#define CAL_KSTA_H 0xE7

//Config register = 0xF5-F6
#define OSC_TRIM_VALUE 0xF7

//Bits within configuration register 0x92
#define POR_TEST 10
class MLX90621 {
private:
    /* Variables */
    byte refreshRate; //Частота обновления
    float temperatures[64]; //Массив температур
    float Tambient; // Отслеживает изменение температуры окружающей среды
датчика
byte loopCount = 0; // используется в основном цикле
    /* Methods */
    void readEEPROM();
    void setConfiguration();
    void writeTrimmingValue();
    void calculateTA();
    void readPTAT();
    void calculateTO();
```

```
        void readIR(); // считываются начальные значения, полученные камерой.
//Затем они вычитаются из измеренных значений.
        void readCPIX();
        void preCalculateConstants();
        int16_t twos_16(uint8_t highByte, uint8_t lowByte);
        int8_t twos_8(uint8_t byte);
        uint16_t unsigned_16(uint8_t highByte, uint8_t lowByte);
        uint16_t readConfig();
        boolean checkConfig();
        float v_ir_off_comp, ksta, v_ir_tgc_comp, v_ir_comp, alpha_comp;
        float tak4, resolution_comp;
        int16_t a_common, a_i_scale, b_i_scale, k_t1_scale, k_t2_scale, resolution;
        uint8_t eepromData[256]; // the full EEPROM reading from the MLX90621
        float k_t1, k_t2, emissivity, tgc, alpha_cp, a_cp, b_cp, v_th;
        uint16_t ptat;
        int16_t cpix;
        float a_ij, b_ij, alpha_ij;
        float minTemp, maxTemp;
    public:
        int16_t irData[64]; //необработанные данные с сенсора
        void initialise(int);
        void measure(bool);
        float getTemperature(int num);
        float getAmbient();
        float getMinTemp();
        float getMaxTemp();
    };
    #endif

    #endif
```

```
//основной файл MLX90621.cpp

#include "MLX90621.h"

void MLX90621::initialise(int refrate) {
    refreshRate = refrate;
    Wire.begin();
    delay(5);
    readEEPROM();
    writeTrimmingValue();
    setConfiguration();
    preCalculateConstants();
}

void MLX90621::measure(bool calculate_temps) {
    if (checkConfig()) {
        readEEPROM();
        writeTrimmingValue();
        setConfiguration();
    }
    readPTAT();
    readIR();
    if(calculate_temps){
        calculateTA();
        readCPIX();
        calculateTO();
    }

}
```

```
float MLX90621::getTemperature(int num) {
    if ((num >= 0) && (num < 64)) {
        return temperatures[num];
    } else {
        return 0;
    }
}

float MLX90621::getAmbient() {
    return Tambient;
}

void MLX90621::setConfiguration() {
    byte Hz_LSB;
    switch (refreshRate) {
    case 0:
        Hz_LSB = 0b00111111;
        break;
    case 1:
        Hz_LSB = 0b00111110;
        break;
    case 2:
        Hz_LSB = 0b00111101;
        break;
    case 4:
        Hz_LSB = 0b00111100;
        break;
    case 8:
        Hz_LSB = 0b00111011;
        break;
```

```
        case 16:
                Hz_LSB = 0b00111010;
                break;
        case 32:
                Hz_LSB = 0b00111001;
                break;
        default:
                Hz_LSB = 0b00111110;
        }
        byte defaultConfig_H = 0b01000110;  //kmoto: See data sheet p.11 and 25
        Wire.beginTransmission(0x60);
        Wire.write(0x03);
        Wire.write((byte) Hz_LSB - 0x55);
        Wire.write(Hz_LSB);
        Wire.write(defaultConfig_H - 0x55);
        Wire.write(defaultConfig_H);
        Wire.endTransmission();

        //Read the resolution from the config register
        resolution = (readConfig() & 0x30) >> 4;
    }

    void MLX90621::readEEPROM() { // Read in blocks of 32 bytes to accomodate Wire library
      for(int j=0;j<256;j+=32) {
        Wire.beginTransmission(0x50);
        Wire.write(j);
        byte rc = Wire.endTransmission(false);
        Wire.requestFrom(0x50, 32);
        for (int i = 0; i < 32; i++) {
```

```
        eepromData[j+i] = (uint8_t) Wire.read();
      }
    }
  }

  void MLX90621::writeTrimmingValue() {
      Wire.beginTransmission(0x60);
      Wire.write(0x04);
      Wire.write((byte) eepromData[OSC_TRIM_VALUE] - 0xAA);
      Wire.write(eepromData[OSC_TRIM_VALUE]);
      Wire.write(0x56);
      Wire.write(0x00);
      Wire.endTransmission();
  }

  void MLX90621::calculateTA(void) {
      Tambient = ((-k_t1 + sqrt(sq(k_t1) - (4 * k_t2 * (v_th - (float) ptat))))
                  / (2 * k_t2)) + 25.0;
  }

  void MLX90621::preCalculateConstants() {
      resolution_comp = pow(2.0, (3 - resolution));
      emissivity = unsigned_16(eepromData[CAL_EMIS_H],
eepromData[CAL_EMIS_L]) / 32768.0;
      a_common = twos_16(eepromData[CAL_ACOMMON_H],
eepromData[CAL_ACOMMON_L]);
      a_i_scale = (int16_t)(eepromData[CAL_AI_SCALE] & 0xF0) >> 4;
      b_i_scale = (int16_t) eepromData[CAL_BI_SCALE] & 0x0F;

      alpha_cp = unsigned_16(eepromData[CAL_alphaCP_H],
```

```
eepromData[CAL_alphaCP_L]) /
                    (pow(2.0, eepromData[CAL_A0_SCALE]) * resolution_comp);
    a_cp = (float) twos_16(eepromData[CAL_ACP_H], eepromData[CAL_ACP_L]) / resolution_comp;
    b_cp = (float) twos_8(eepromData[CAL_BCP]) / (pow(2.0, (float)b_i_scale) * resolution_comp);
    tgc = (float) twos_8(eepromData[CAL_TGC]) / 32.0;

    k_t1_scale = (int16_t) (eepromData[KT_SCALE] & 0xF0) >> 4;
    k_t2_scale = (int16_t) (eepromData[KT_SCALE] & 0x0F) + 10;
    v_th = (float) twos_16(eepromData[VTH_H], eepromData[VTH_L]);
    v_th = v_th / resolution_comp;
    k_t1 = (float) twos_16(eepromData[KT1_H], eepromData[KT1_L]);
    k_t1 /= (pow(2, k_t1_scale) * resolution_comp);
    k_t2 = (float) twos_16(eepromData[KT2_H], eepromData[KT2_L]);
    k_t2 /= (pow(2, k_t2_scale) * resolution_comp);
}

void MLX90621::calculateTO() {
    float v_cp_off_comp = (float) cpix - (a_cp + b_cp * (Tambient - 25.0));
    tak4 = pow((float) Tambient + 273.15, 4.0);
    minTemp = NULL, maxTemp = NULL;
    for (int i = 0; i < 64; i++) {
        a_ij = ((float) a_common + eepromData[i] * pow(2.0, a_i_scale)) / resolution_comp;
        b_ij = (float) twos_8(eepromData[0x40 + i]) / (pow(2.0, b_i_scale) * resolution_comp);
        v_ir_off_comp = (float) irData[i] - (a_ij + b_ij * (Tambient - 25.0));
        v_ir_tgc_comp = (float) v_ir_off_comp - tgc * v_cp_off_comp;
        float alpha_ij = ((float) unsigned_16(eepromData[CAL_A0_H],
```

```
eepromData[CAL_A0_L]) / pow(2.0, (float) eepromData[CAL_A0_SCALE]));
            alpha_ij += ((float) eepromData[0x80 + i] / pow(2.0, (float)
eepromData[CAL_DELTA_A_SCALE]));
            alpha_ij = alpha_ij / resolution_comp;
            //ksta = (float) twos_16(eepromData[CAL_KSTA_H],
eepromData[CAL_KSTA_L]) / pow(2.0, 20.0);
            //alpha_comp = (1 + ksta * (Tambient - 25.0)) * (alpha_ij - tgc * alpha_cp);
            alpha_comp = (alpha_ij - tgc * alpha_cp);  // For my MLX90621 the
ksta calibrations were 0
                                                                                //
so I can ignore them and save a few cycles
            v_ir_comp = v_ir_tgc_comp / emissivity;
            float temperature = pow((v_ir_comp / alpha_comp) + tak4, 1.0 / 4.0) -
274.15;

            temperatures[i] = temperature;
            if (minTemp == NULL || temperature < minTemp) {
                minTemp = temperature;
            }
            if (maxTemp == NULL || temperature > maxTemp) {
                maxTemp = temperature;
            }
        }
    }

    float MLX90621::getMinTemp() {
        return minTemp;
    }

    float MLX90621::getMaxTemp() {
```

```
    return maxTemp;
}


void MLX90621::readIR() {
    for (int j = 0; j < 64; j += 16) { // Read in blocks of 32 bytes to overcome Wire buffer limit
        Wire.beginTransmission(0x60);
        Wire.write(0x02);
        Wire.write(j);
        Wire.write(0x01);
        Wire.write(0x20);
        Wire.endTransmission(false);
        Wire.requestFrom(0x60, 32);
        for (int i = 0; i < 16; i++) {
            uint8_t pixelDataLow = (uint8_t) Wire.read();
            uint8_t pixelDataHigh = (uint8_t) Wire.read();
            irData[j + i] = twos_16(pixelDataHigh, pixelDataLow);
        }
    }
}

void MLX90621::readPTAT() {
    Wire.beginTransmission(0x60);
    Wire.write(0x02);
    Wire.write(0x40);
    Wire.write(0x00);
    Wire.write(0x01);
    Wire.endTransmission(false);
    Wire.requestFrom(0x60, 2);
```

```
    byte ptatLow = Wire.read();
    byte ptatHigh = Wire.read();
    ptat = (ptatHigh * 256) + ptatLow;

}

void MLX90621::readCPIX() {
    Wire.beginTransmission(0x60);
    Wire.write(0x02);
    Wire.write(0x41);
    Wire.write(0x00);
    Wire.write(0x01);
    Wire.endTransmission(false);
    Wire.requestFrom(0x60, 2);
    byte cpixLow = Wire.read();
    byte cpixHigh = Wire.read();
    cpix = twos_16(cpixHigh, cpixLow);
}

int16_t MLX90621::twos_16(uint8_t highByte, uint8_t lowByte){
    uint16_t combined_word = 256 * highByte + lowByte;
    if (combined_word > 32767)
        return (int16_t) (combined_word - 65536);
    return (int16_t) combined_word;
}

int8_t MLX90621::twos_8(uint8_t byte) {
    if (byte > 127)
        return (int8_t) byte - 256;
    return (int8_t) byte;
```

```
}

uint16_t MLX90621::unsigned_16(uint8_t highByte, uint8_t lowByte){
    return (highByte << 8) | lowByte;
}

uint16_t MLX90621::readConfig() {
    Wire.beginTransmission(0x60);
    Wire.write(0x02);
    Wire.write(0x92);
    Wire.write(0x00);
    Wire.write(0x01);
    Wire.endTransmission(false);
    Wire.requestFrom(0x60, 2);
    byte configLow = Wire.read();
    byte configHigh = Wire.read();
    uint16_t config = ((uint16_t) (configHigh << 8) | configLow);
    return config;
}


// Опрос MLX90621 текущий статус
  //Returns true if the POR/Brown out bit is set
  boolean MLX90621::checkConfig() {
    bool check = !((readConfig() & 0x0400) >> 10);
    return check;
  }
```