**СОФИЙСКА ПРОФЕСИОНАЛНА ГИМНАЗИЯ ПО ЕЛЕКТРОНИКА „ДЖОН АТАНАСОВ“**

**Д И П Л О М Е Н П Р О Е К Т**

**Тема ““**

**Практическа част: „“**

Дипломант: Кристиан Николаев Димитров 12в клас

Професия: TODO(481030) „ Приложен програмист“

Специалност: TODO(4810301) „Приложно програмиране“

Ръководител на дипломен проект: инж. Любица Димитрова

Дипломант:

*/подпис/*

Ръководител на Дипломен проект:

*/подпис/*

София

2022

**Списък на използваните термини и съкращения**

Timelapse - техника в фотографията, при която се правят множество снимки през определен интервал от време.

Hyperlapse - вид timelapse, при който камерата бива премествана в пространството след всеки заснет кадър.

IDE (интегрирана среда за разработка) - софтуерно приложение, което предоставя цялостна среда за разработване на софтуер на програмистите.

<https://bg.wikipedia.org/wiki/Интегрирана_среда_за_разработка>

C++ - програмен език от високо ниво, който се използва за разработване на различни видове приложения, включително операционни системи, игри и друг софтуер с висока производителност. Той е разработен от Бярне Страуструп в началото на 1980-те години като разширение на програмния език C. C++ е известен със своята ефективност, гъвкавост и възможностите за обектно-ориентирано програмиране (ООП), позволявайки на разработчиците да създават преизползваем код и да проектират сложни софтуерни системи.

Стъпков мотори - още наричан стъпков двигател е вид безколекторен постояннотоков двигател, който разделя пълното завъртане на няколко равни стъпки. Моторът може да се позиционира чрез сензор или друга обратна връзка (Closed Loop Control System). Той не може да бъде свързван директно към електрическа мрежа. Управлението на стъпковите електродвигатели се извършва само чрез компоненти на електрониката, като съществува голямо разнообразие на схеми и елементи за реализацията им, модули за вграждане и самостоятелни прибори.

<https://bg.wikipedia.org/wiki/Стъпков_двигател>

Входните/изходните пинове -

I2C - Тази комуникация е предназначена да позволи на множество "подчинени ИС" да комуникират с един или повече "главни ИС". Протоколът е предназначение за комуникация само на къси разстояния в рамките на едно устройство или между две свързани устройства. Изисква само два сигнални проводника за обмен на информация. I2C шината се състой от два сигнала: SCL и SDA. SCL е тактовият сигнал(часовника), а SDA е сигнала за данни. I2C има 7-битово адресно пространство с рядко използвано 10-битово разширение. Според това дали даден бит е за четене или писане (read/write) се определя какво действие ще прави главното устройство и съответно подчиненото, кое ще предава и кое ще получава данни.

<https://www.electronicevolution.bg/bg-news-details-46.html>

*Master устройство -*

STOP сигнал -

JSON -

***Теоретична част***

***Увод***

*Живеем във време, в което дигиталните и механични устройства за автоматизация са неизменна част от живота ни. Подобни технологии спестяват време и усилия от страна на хората, както и намаляват разходи. Друго предимство на технологиите за автоматизация, към което разработчиците се стремят, е удобният за потребителя интерфейс. С нарастването на смартфоните и другите устройства технологията стана по-достъпна и по-лесна за използване от хора от всички възрасти и произход.*

*Фотографията и видеографията играят важна роля в социалните мрежи, телевизионната и рекламна индустрия, защото визуалното представяне е първото нещо, от което зависи дали вниманието на зрителите ще се задържи.*

*Като за начало, нека кажем какво е timelapse фотографията. Това е метод, при който фотограф прави поредица от снимки на предварително определен интервал за определена продължителност и след това представя тези снимки със скорост, по-бърза от реално време, под формата на видеоклип. За първи път тази техника е използвана от френския режисьор Жорж* Мелиес през 1897 г.

Hyperlapse фотографията, която целим да улесним, е по-модерната и грабваща окото техника. Hyperlapse, всъщност е подвижен timelapse. Тоест между отделните снимки се извършва леко движение на камерата, като по този начин се придава дълбочинен ефект и по-голяма динамика на крайното видео. Проблемът при тази техника е, че за да се създаде подобен видеоклип е необходимо голямо усилие от страна на фотографа. Причината за това е, че неговата работа се състои в това да премества статива с камера, след всеки заснет кадър, което понякога може да продължава с часове.

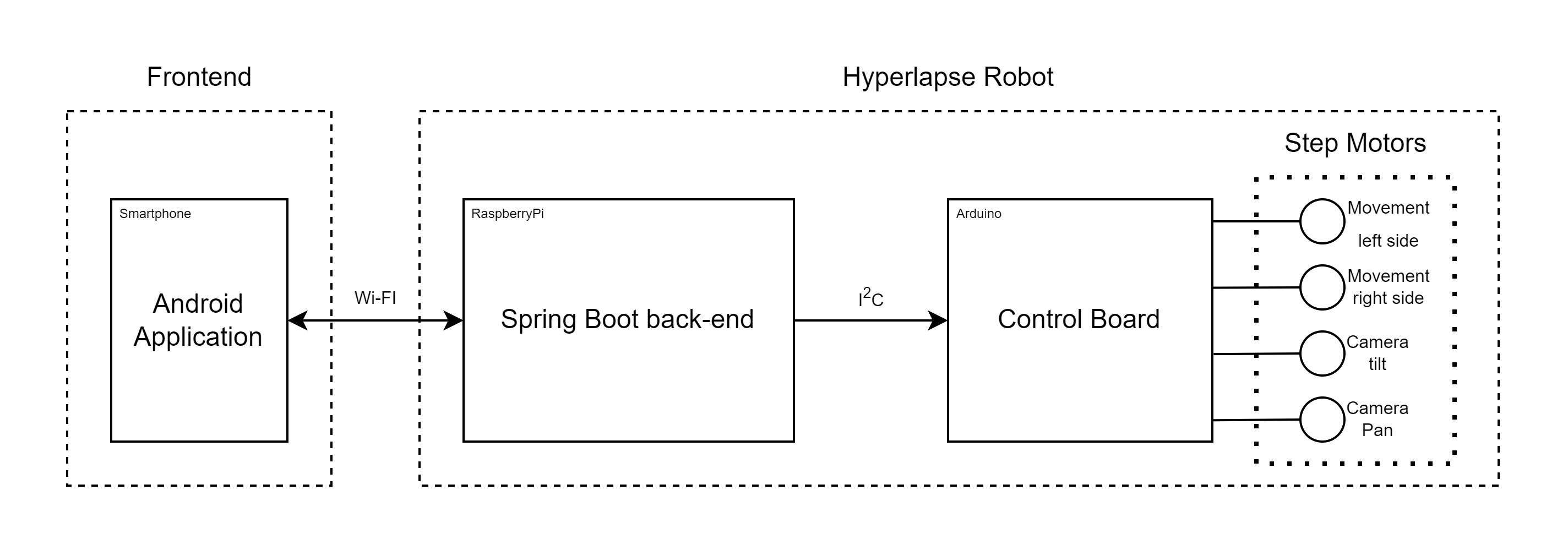
TODO Къде би могъл да се приложи?

Цел на проекта:

Да се улесни процеса за заснемане на Hyperlapse видеоклипове, чрез автоматизирано решение. Да се създаде робот, който да може да се движи и да контролира посоката камера, предназначена за заснемане, като задаването на тези инструкции става през мобилно приложение, чрез отдалечен достъп към робота.

**Изложение**

**Глава I. Анализ**

***Фиг. 1. Софтуерна архитектура на проекта***

С напредването на технологиите, микрокомпютрите и логически контролери стават все по-достъпни, поради масовото им производство. Едно от предимствата на тези устройства е, че те предразполагат лесното изграждането на прототипи или дори завършени системи. В роботиката, те са главните структурни единици, които управляват съответната машина.

Темата, която засягаме е електрически робот, който автоматизира процеса на заснемане на hyperlapse. За целта, той ще трябва да може да извършва движение в пространството, както и да контролира посоката на където да снима камерата. За да бъде навигиран от потребител, роботът има функция за управление от отдалечено разстояние.

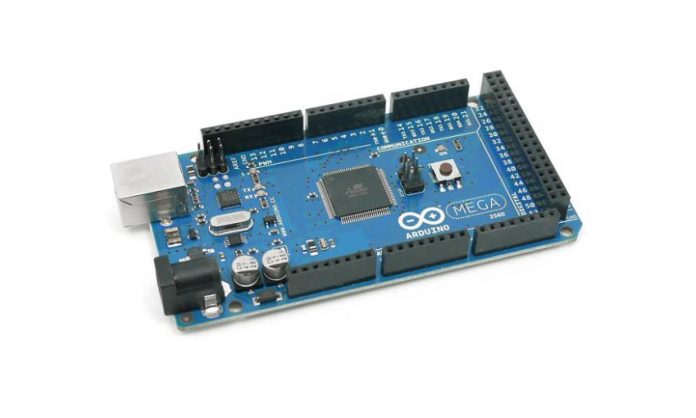
За да може да се изработи такъв робот е необходима подготовка и изследване при изборът на компоненти. Важно е предварително да се предвижда дали те ще са съвместими един с друг, както и да няма твърде голяма консумация на електроенергия, за да се постигне по-голяма ефективност.

Проектът се разделя на няколко основни комуникиращи помежду си компонента, за да може да предава съществената информация, която засяга всеки един от тях. Нека разгледаме всеки един от тях, за да добием представа за архитектурата.

**1.1 Първи компонент - Arduino**

**1.1.1 Какво е Arduino?** - представлява хардуерна платка, която се състои от микроконтролер, явяващ се и микро компютър. Технологията е изградена с идеята да може да се използва лесно за софтуерни и хардуерни проекти. Компанията ,,Arduino LLC” се е погрижила за създаването на IDE, което позволява потребителите да разработват, компилират и качват код на Arduino платка. Програмният език, с който работят платките е опростена версия на C++. Тези устройства имат различни входове и изходи, които могат да се използват за свързване към сензори, електродвигатели, светлини и други електронни компоненти.

**1.1.2 Каква е функцията на Arduino?** - в нашия проект, Arduino ще се занимава с изпълнението на логиката, стояща зад контролирането на задвижващите и управляващите камерата стъпкови мотори. То също има функцията да приема инструкции за изпълнение, които получава от по-горните слоеве на архитектурата.

**1.1.3 Кой модел Arduino използваме?** - поради голямото разнообразие от различни видове модели Arduino платки, трябва да се вземе под внимание кой от тях би бил годен за работа в нашия робот. При избора на Arduino има няколко фактора, които трябва да се вземат под внимание:

* Брой и тип на входните/изходните пинове - Трябва да се уверим, че платката разполага с достатъчно входни/изходни пинове, за да можем да свържем всички необходими компоненти за робота.
* Процесорна мощност - Ако роботът изисква бърза обработка на данни или изчисления, ще имаме нужда от по-мощна платка с по-бърз процесор.
* Размер и форма на платката - трябва да направим сметка, дали размерът на платката е съвместима с конструкцията на робота.

Относно целите на нашият проект, Arduino Mega 2560 е за предпочитание, защото има повече пинове и по-голяма процесорна мощност.

**1.1.4 Как Arduino изпълнява своите задачи? -** когато платката бива захранвана с електричество, тя автоматично започва изпълнението на предварително каченият програмен код, който сме разработили. Като за начало се инициализират изходните пинове, чрез които ще се управляват стъпковите мотори. Създава се и I2C връзка, която позволява на Arduino да комуникира с други електронни компоненти (в нашият случай RaspberryPI). След това то започва да чака информация от master устройството, като данните идват на порции от по 8 байта. Това ще се случва докато не се получи STOP сигнал (в случая символа‘$’, който в двоичен код е ‘00100100’). Порциите от байтове се събират в един стринг, чийто формат се очаква да бъде JSON, но това не зависи от master устройството (понеже то изпраща данните). Този JSON стринг представлява поредица от инструкции, които се отнасят до управлението на моторите. За да се намали обема на информация, която се предава в този JSON стринг сме направили съкращения в имената на отделните полета с цел за по-бързо получаване на данните по I2C връзката.

Нека разгледаме един пример за това какво съдържа един JSON стринг с инструкция:

|  |  |
| --- | --- |
| **Съкратен вариант, който Arduino използва.** | **Не съкратен вариант за по-добро възприемане на произхода на съкращенията.** |
| {"rc":1,"wr":1.166666,"r":[{"lm":{"ds":100.0,"t":60.0},"rm":{"ds":100.0,"t":60.0},"pm":{"dg":50.0,"t":60.0},"tm":{"dg":-20.0,"t":60.0}}]} | {  "rulesCount":1,  "wheelRadius":1.166666,  "rules":[  {  "leftMotor":{  "distance":100.0,  "executionTime":60.0  },  "rightMotor":{  "distance":100.0,  "executionTime":60.0  },  "panMotor":{  "degree":50.0,  "executionTime":60.0  },  "tiltMotor":{  "degree":-20.0,  "executionTime":60.0  }  }  ]  } |

Този JSON обект има следните полета:

* “rc” / "rulesCount": Число, което указва броя на инструкциите, които се съдържат в списъка “r” / “rules”. В примера е демонстриран списък със само една инструкция.
* “wr” / "wheelRadius": Число, което представлява радиуса на задвижващите колела на робота в сантиметри. В случая радиусът на колелата е зададен на 1.166666 см.
* “r” / "rules": Списък от обекти, описващ поредица от инструкции. В този случай има само една инструкция в списъка. Всеки обект “r” / “rules” има следните свойства:
  + "leftMotor": Обект, който съдържа информация за стъпковият мотор, задвижващ лявото колело на робота. Това включва разстоянието (“ds” / “distance”), което моторът трябва да измине в сантиметри (в случая 100.0 см) и времето за изпълнение (“t” / “executionTime”) на това разстояние в секунди (в случая 60.0 сек).
  + "rightMotor": Обект, който съдържа информация за стъпковият мотор, задвижващ дясното колело на робота. Това включва разстоянието (“ds” / “distance”), което моторът трябва да измине в сантиметри (в случая 100.0 см) и времето за изпълнение (“t” / “executionTime”) на това разстояние в секунди (в случая 60.0 сек).
  + "panMotor": Обект, който съдържа информация за стъпковият мотор, контролиращ хоризонталната посока на камерата на робота. Това включва градусната мярка (“dg” / “degree”), до която моторът трябва да се завърти (в случая 50.0 градуса спрямо сегашната позиция) и времето за изпълнение (“t” / “executionTime”) на това завъртане в секунди (в случая 60.0 сек).
  + "tiltMotor": Обект, който съдържа информация за стъпковият мотор, контролиращ вертикалната посока на камерата на робота. Това включва градусната мярка (“dg” / “degree”), до която моторът трябва да се завърти (в случая 20.0 градуса спрямо сегашната позиция) и времето за изпълнение (“t” / “executionTime”) на това завъртане в секунди (в случая 60.0 сек).

Функционалност за завиване

1. Как ще се извършва завиването на робота?

Робота се движи посредством 2 стъпкови мотора, прикрепени от двете му страни. Чрез едновременно завъртане на моторите с различни скорости, би могло да се постигне ефект на завиване. За да се изчисли скоростта, трябва да се вземат в предвид разстоянието (което конкретният мотор трябва да измине) и времето за изпълнение.

1. Изчисляване на данните за извършване на завой.

* Като за начало е важно да се определят какви ще бъдат входните данни, които ще бъдат използвани за изчислението, както и изходните данни, които ще се изпращат, като инструкция към робота.

**Входни данни (задаващи се от потребителя през мобилното управляващо приложение):**

radius (cm)- радиус на завоя.

angle - ъгъл на завиване.

execution time (sec) - време на изпълнение.

axle track (cm) - дължината на оста между двете задвижващи колела. (Този параметър е зададен предварително, тъй като той е част от конструкцията на робота).

**Изходни данни:**

left motor distance (cm) - разстоянието, което левият мотор трябва да измине.

left motor execution time (sec) - времето на изпълнение на левият мотор.

right motor distance (cm) - разстоянието, което десният мотор трябва да измине.

right motor execution time (sec) - времето на изпълнение на десният мотор.

|  |  |
| --- | --- |
| На диаграмата е демонстрирано как входните и изходните данни могат да се представят в графичен вид.  radius (cm) - червената окръжност описва пътят по когото роботът ще се движи. Центърът на тази окръжност се намира на *radius* cm спрямо центъра на задвижващата ос (axle track / 2).  angle - ъгълът на завой с връх стоящ в центъра на окръжността описваща пътя на движение, като едното рамо на ъгъла съвпада с продължението на оста за движение. |  |
| Left motor distance (cm) - разстоянието, което левият мотор трябва да измине се изразява, чрез дъгата започваща от лявото колело и завършваща при точката left motor target position от лилавата окръжност.  Формула:  *leftMotorPathRadius*  *leftMotorDistance*  right motor distance (cm) - разстоянието, което десният мотор трябва да измине се изразява, чрез дъгата започваща от дясното колело и завършваща при точката right motor target position от синята окръжност.  Формула:  *rightMotorPathRadius*  *rightMotorDistance* | |