Programska implementacija sustava za praćenje i vizualizaciju stanja i statusa uređaja baziranih na Arduino platformi

Borović, Domagoj

Sveučilište
Sjever

Završni rad br. 363/EL/2015

Programska implementacija sustava za praćenje i vizualizaciju stanja i statusa uređaja

Domagoj Borović, 4092/601

Varaždin, rujan 2015.

Sveučilište
Sjever

Odjel za Elektrotehniku

Završni rad br. 363/EL/2015

Programska implementacija sustava za praćenje i vizualizaciju stanja i statusa uređaja

Student

Domagoj Borović, 4092/601

Mentor

mr. sc. Matija Mikac, viši predavač

Varaždin, rujan 2015.

Predgovor

Završni rad posvećujem svojoj obitelji koja je sve ove godine bila uz mene te mi pružala punu potporu. Zahvaljujem im se na strpljenju te kako psihičku tako i financijsku podršku u studiranju. Bez njih ne bih nikada došao do ovog stadija studiranja.

Također zahvaljujem i kolegama sa kojima sam svakodnevno komunicirao o važnim stvarima o nastavi te pripremama za ispite.

Na kraju se želim zahvaliti svim profesorima na trudu kako na predavanjima tako i na konzultacijama van predavanja.

Sažetak

U ovom radu obrađena je Arduino platforma, njezine mogućnosti te hardverski i softverski dio iste. Nadalje, obrađen je senzor težine (eng. *Load cell*) kojem se mijenja otpor ovisno o naprezanju. Također je opisan i HX711 24 bitni analogno-digitalni pretvornik te Arduino *Ethernet Shield* kojim Arduino dobiva pristup Internetu te mogućnost pisanja i čitanja na SD (eng. *Secure Digital*) karticu. Opisana je izrada web poslužitelja sa jednostavnom MySQL bazom podataka te komunikacija izrađenog uređaja sa poslužiteljem, spremanje podataka u bazu te prikaz rezultata na web stranici. Također je opisan zapis podataka na SD karticu u .csv (eng. *Comma Separated Value*) formatu i čitanje istog pomoću Microsoft Excel aplikacije. Obrađeno je i dijeljenje sadržaja SD kartice preko Interneta te pristupanje .csv datoteci koja se nalazi na njoj.

Ključne riječi: Arduino, senzor težine, HX711, Arduino *Ethernet Shield*, web poslužitelj, baza podataka, web stranica, Excel aplikacija.

Sadržaj

1. Uvod

Pomoću Arduino platforme otvorenog koda (eng. *Open Source*) mogu se realizirati razni uređaji koji se koriste u proizvodnji, nadzoru te ostalim dijelovima postrojenja. Osim sklopovlja, platforma uključuje kompletno razvojno okruženje koje omogućava jednostavan razvoj i programiranje u programskom jeziku C/C++. Pomoću nje vrlo je lako realizirati gotovo svaku ideju koja je potrebna korisniku. Mana ove platforme je ograničenje memorije. Problem memorije se rješava „jačim“ Arduino modelima jer Uno je osnovni model koji je namijenjen prvenstveno za početnike odnosno za učenje rada sa mikrokontrolerima.

Ovoj platformi mogu se dodavati razni moduli poput *Ethernet Shielda* za pristup Internetu te SD kartici, *bluetooth* modul koji omogućava komunikaciju preko *bluetootha* itd. Prednost ovog sučelja je u tome što ga koristi puno programera pa su knjižnice vrlo rasprostranjene te ako programer zapne na nekom dijelu koda može postaviti pitanje za pomoć na Arduino forumu na kojem ima veliki broj korisnika pa se na odgovor za pomoć ne treba dugo čekati.

Ideja završnog rada je izraditi digitalnu vagu pomoću Arduino platforme otvorenog koda koja će korisniku omogućiti da u svakom trenutku može saznati težinu objekta koji se nalazi na toj vagi bez da prethodno mora bespotrebno gubiti vrijeme na dolaženje do te vage jer bi ona mogla biti relativno jako udaljena od njegove trenutne pozicije. Korisnik će jednostavno moći upaliti računalo te otvoriti web stranicu u bilo kojem pregledniku te moći vidjeti težinu objekta na njegovoj vagi te vidjeti potrebnu statistiku koja ga zanima prikazanu pomoću grafova.

Izrađeni uređaj ne mora nužno imati pristup Internetu. Zbog toga je napravljena i funkcionalnost zapisa na microSD karticu. Korisnik može npr. na kraju radnog dana uzeti karticu iz uređaja te ju pomoću adaptera ili mobitela prebaciti na računalo. Nakon što ima podatke o mjerenjima, na računalu jednostavno pokrene izrađenu aplikaciju u Excelu te dobije svu statistiku koja mu je potrebna. Osim statistika koje su trenutno napravljene u Excelu može se po zahtjevima korisnika proširiti te staviti još dodatne.

Uz to izvedena je i mogućnost da Arduino dijeli podatke koji se nalaze na njegovoj SD kartici te na web stranici možemo te datoteke vidjeti jednostavnim pritiskom gumba. Za realiziranje ovog rada potrebno je imati Arduino sklop, *Ethernet Shield* za spajanje na Internet, senzor težine te HX711 analogno digitalni pretvornik.

2. Arduino UNO R3

Arduino UNO R3 popularna je razvojna platforma za mnoge programere koji se bave automatizacijom. Postala je popularna zbog jednostavnosti korištenja te velike zastupljenosti knjižnica za pojedine dodatne module koji se mogu spojiti na osnovnu pločicu kao što su senzori, sklopke, releji, *Shieldovi* itd. Ova platforma je kompletno razvojno okruženje tj. hardver + softver. Pogodan je za početnike i profesionalce iz razloga jer se na Internetu može naći puno korisnih informacija o programiranju i implementaciji dodatnih modula. Napajati ga se može računalom preko USB konektora ili zasebnog izvora poput baterije ili ispravljača napona 6-20V. USB konektor ujedno se koristi za komunikaciju s okolinom. Praćenje i nadzor prijenosa podataka moguć je integriranim alatom koji se zove *Serial Monitor.*

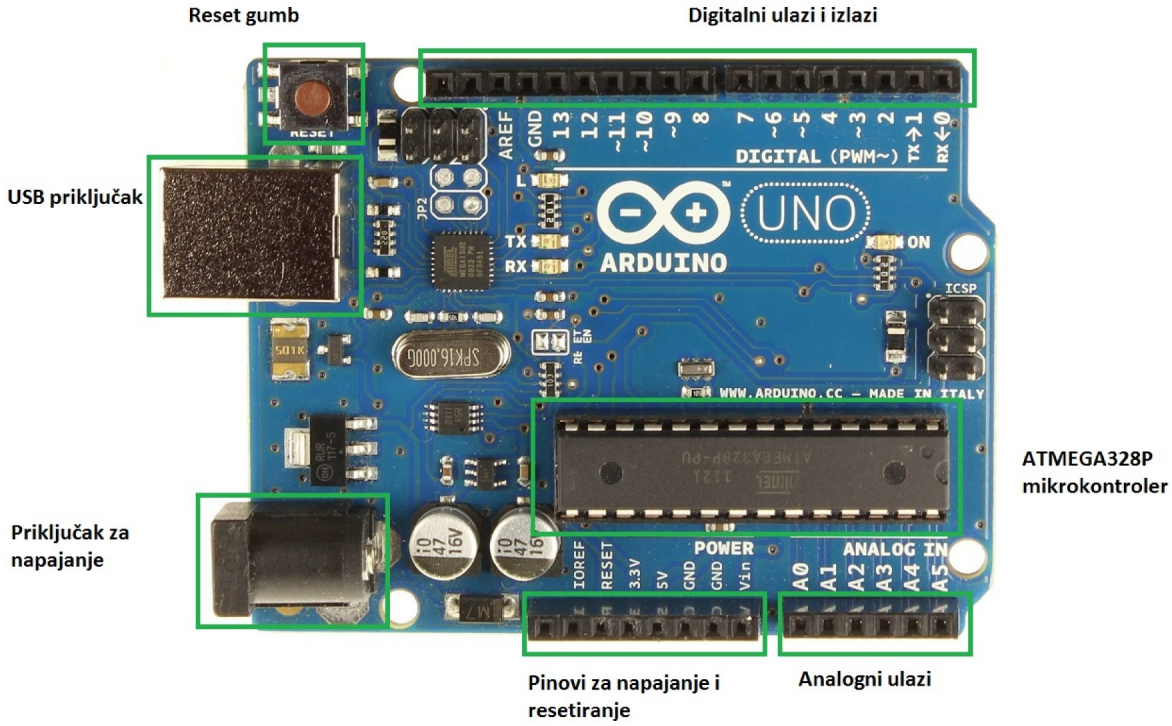
2.1. Tehničke karakteristike i najvažniji dijelovi

Mikrokontroler	Atmega328P
Radni napon mikrokontrolera	5V
Preporučeni ulazni napon	7-12V
Granice ulaznog napona	6-20V
Digitalni ulazno izlazni pinovi	14 (od kojih 6 ima pulsno širinsku modulaciju)
Analogni ulazni pinovi	6
Istosmjerna struja za pinove	20mA
Istosmjerna struja za 3.3V pin	50mA
Flash memorija	32KB (Bootloader koristi 0.5KB)
SRAM memorija	2KB
EEPROM memorija	1KB
Frekvencija oscilatora	16MHz

Tablica 1 - Tehničke karakteristike Uno R3

U tablici 1 nabrojene su najvažnije tehničke karakteristike iz kojih dobijemo uvid u komponente te mogućnosti Arduina. Prema specifikacijama, moramo pripaziti na jakost struje i napone koje dovodimo na pinove kako ne bi premašile one koje Arduinovi pinovi mogu podnijeti jer bi moglo doći do trajnih oštećenja mikrokontrolera zbog pregaranja.

Prema specifikacijama memorije možemo odlučiti koji model Arduina trebamo za projekt.



Slika 1 - Izgled Arduino R3 uređaja [3]

2.2. Dodatna oprema potrebna za realizaciju rada

Za izradu završnog rada potrebna nam je sljedeća dodatna oprema:

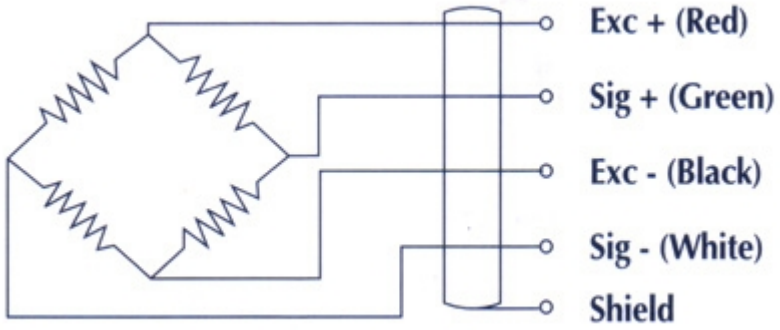
- Senzor težine
- HX711 24 bitni analogno digitalni pretvornik sa pojačalom
- Arduino *Ethernet Shield*
- SD kartica

2.2.1. Senzor težine

U ovom radu korišten je Zemic L6E Load cell senzor težine. Radi na principu piezo-otpornosti. Piezo-otpornost [3] je promjena otpornosti poluvodiča ili metala kada je opterećen mehaničkom silom. On pretvara mehaničku silu u električni signal zbog promjene otpora. Za razliku od piezo-električnog efekta, piezo-otporni efekt mijenja samo električni otpor, a ne i električni potencijal. Promjenom otpora mijenja se i struja koja teče Whaetstonovim mostom u koji su spojeni njegovi promjenjivi piezo-otpornici.

Zemic L6E ima različite raspone mjerenja težina ovisno o modelu. U radu je korišten model koji može mjeriti do 300kg. Napravljen je od legure aluminija te je zaliven silikonom.

Slika 2 - Izgled senzora [3]



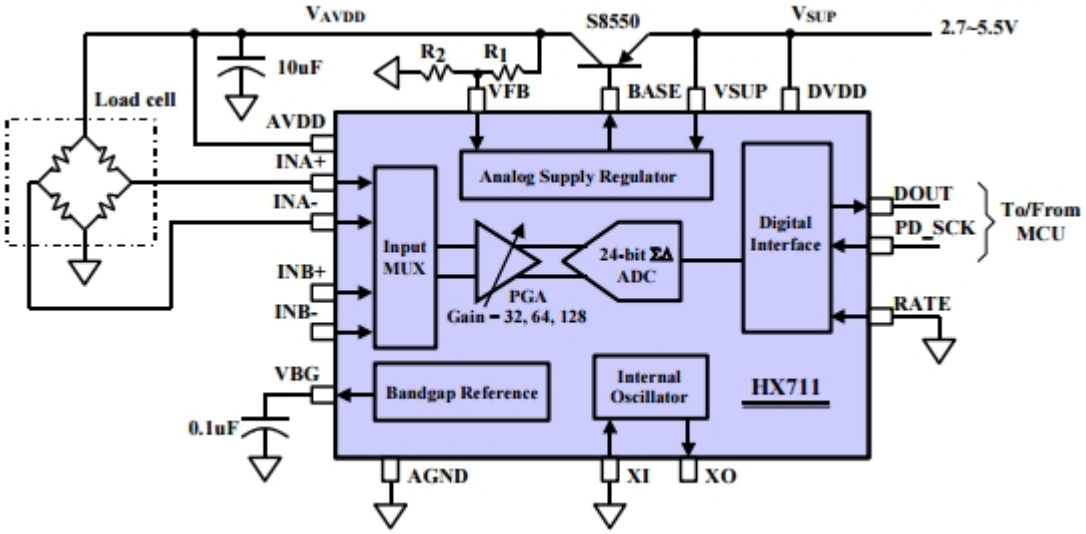
Slika 3 - Izvodi iz senzora [4]

Iz slike 3 je vidljivo da treba spojiti ulazni napon tj. crvenu žicu na + polaritet izvora, a crnu žicu na – polaritet. Izlaz iz senzora su zelena i bijela žica od kojih se zelena spaja na + polaritet, a bijela na – polaritet. Postoji još i *Shield* žica koju treba spojiti na masu.

2.2.2. HX711 analogno digitalni pretvornik

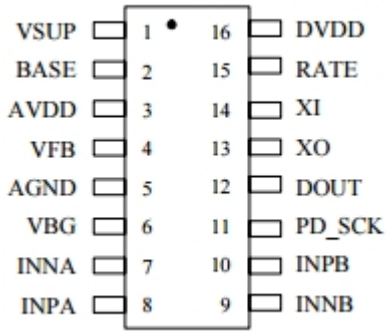
HX711 analogno digitalni pretvornik [2] baziran je na poluvodičkoj tehnologiji. Ovaj precizni 24 bitni analogno digitalni pretvornik dizajniran je specijalno za vage te ostale senzore koji rade na principu mostova poput Whaetstoneovog mosta. Analogni ulaz ima 2 kanala, A i B. A kanal se može programirati da pojačava signal 64 ili 128 puta ovisno o tome kako

isprogramiramo pojačalo kada je na senzor spojeno napajanje od 5 volti. Pojačanju od 64 puta odgovara diferencijalni napon pune skale ±40mV, a pojačanju 128 puta odgovara diferencijalni napon pune skale od ±20mV. B kanal ima fiksno pojačanje od 32 puta. Čip u sebi ima regulator napona tako da nema potrebe spajati eksterni regulator. Unutarnje registre nije potrebno programirati već se cijela kontrola obavlja preko pinova.



Slika 4 - Blok dijagram spajanja senzora na HX711 [2]

Na slici 4 prikazano je spajanje senzora spojenog u Whaetstoneov most na A kanal analogno digitalnog pretvornika. Iz slike je vidljivo da signal prvo ide na multipleksor koji služi za slanje velikog broja signala na pojačalo koje tada signal pojačava ovisno o tome kako je programirano (u većini slučaja 128 puta) te tada taj pojačani signal ide na 24 bitni analogno digitalni pretvornik. Na kraju se taj digitalni signal šalje na izlaz preko pinova serijskog sučelja DOUT i PD_SCK. Kada podaci nisu spremni za preuzimanje iz digitalnog sučelja pin digitalnog izlaza DOUT je postavljen u logičku 1, a pin serijskog takta PD_SCK postavljen je u logičku 0. Kada se pin DOUT postavi u logičku 0 to je indikator da su podaci spremni za prijenos. Ako želimo podatke preuzeti tada je potrebno poslati 25-27 pozitivna impulsa na pin PD_SCK te se tada podaci prenose kroz pin DOUT . Svaki podatak na DOUT se prenosi bit po bit počevši od bita najveće važnosti MSB (eng. *Most Significant bit*). Zadnji impuls na PD_SCK ulazu vraća stanje DOUT pina u logičku 1 što znači da podaci nisu dostupni za prijenos.

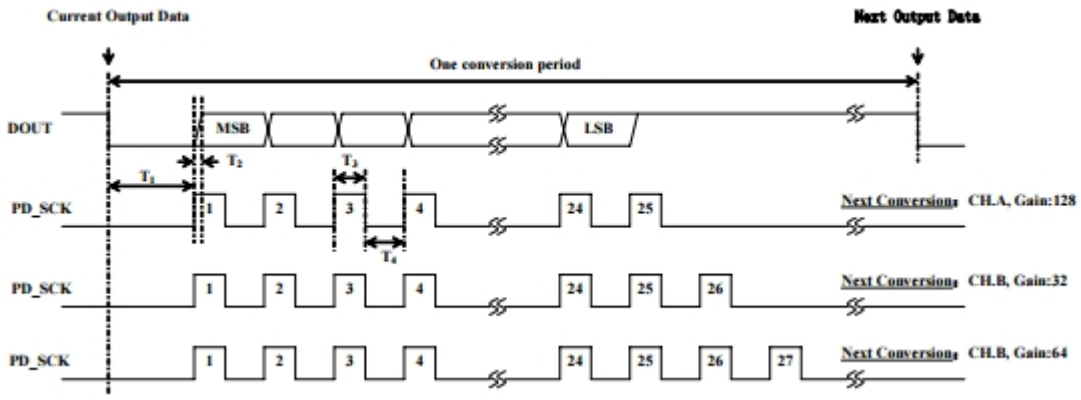


Slika 5 - Pinovi HX711 čipa [2]

PIN	IME PINA	FUNKCIJA	OPIS
1	VSUP	IZVOR NAPONA	REGULATOR IZVORA 2.7-5.5V
2	BASE	ANALOGNI IZLAZ	REGULATOR KONTROLNOG IZLAZA
3	AVDD	IZVOR NAPONA	IZVOR ZA ANALOGNE ULAZE I IZLAZE 2.6-5.5V
4	VFB	ANALOGNI ULAZ	REGULATOR KONTROLNOG ULAZA
5	AGND	MASA	MASA ZA ANALOGNE ULAZE I IZLAZE
6	VBG	ANALOGNI IZLAZ	REFERENTNI BYPASS IZLAZ
7	INNA	ANALOGNI ULAZ (-)	NEGATIVNI POL A KANALA
8	INPA	ANALOGNI ULAZ (+)	POZITIVNI POL A KANALA
9	INNB	ANALOGNI ULAZ (-)	NEGATIVNI POL A KANALA
10	INPB	ANALOGNI ULAZ (+)	POZITIVNI POL A KANALA
11	PD_SCK	DIGITALNI ULAZ	ULAZ ZA KONTROLU
12	DOUT	DIGITALNI IZLAZ	SERIJSKI PODATKOVNI IZLAZ
13	XO	DIGITALNI ULAZ/IZLAZ	KRISTAL ULAZ IZLAZ
14	XI	DIGITALNI ULAZ	KRISTAL ULAZ IZLAZ ILI VANJSKI GENERATOR TAKTA
15	RATE	DIGITALNI ULAZ	KONTROLA BRZINE PRIJENOSA PODATAKA (0: 10Hz; 1: 80Hz)
16	DVDD	IZVOR NAPONA	IZVOR ZA DIGITALNE ULAZE I IZLAZE 2.6-5.5V

Tablica 2 - Opis pinova HX711 čipa

Iz slike 5 i njenog opisa u tablici 2 dobivamo najvažnije podatke o građi ovog modula. Prema tim podacima saznajemo kako se spaja i kontrolira rad tog sklopa.

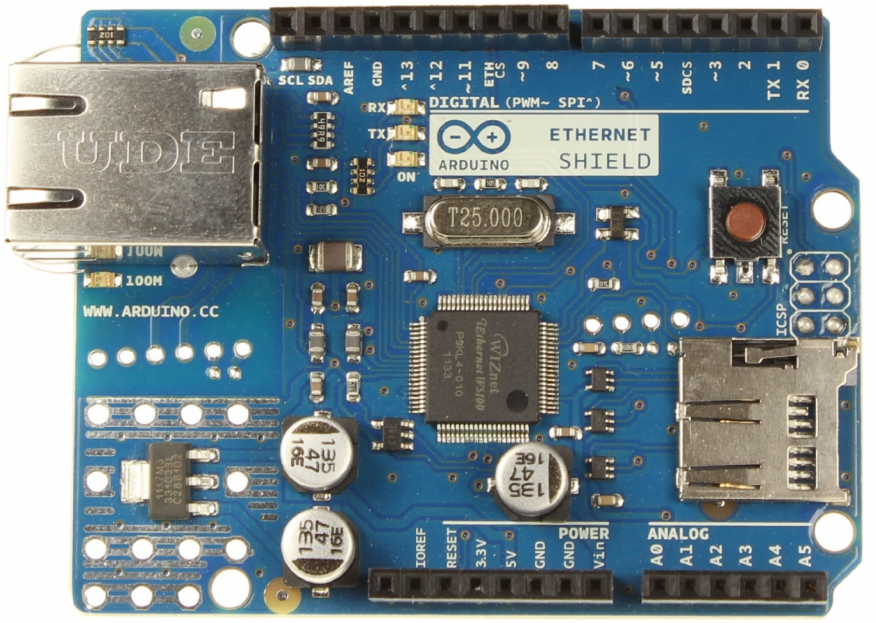


Slika 6 - Vremenski dijagram slanja podataka [2]

Iz dijagrama na slici 6 saznajemo na koji način se šalju podaci iz modula. Vidimo da ako imamo pojačanje 128 puta tada 25. takt na PD_SCK pinu znači da je podatak poslan te se postavlja stanje pina DOUT u logičku 1. Ako imamo pojačanje 64 puta tada je taj takt 27. po redu, a za pojačanje 32 puta 26. po redu.

2.2.3. Arduino *Ethernet Shield*

Arduino *Ethernet Shield* je modul kojim se osnovnom uređaju daje mogućnost spajanja na Internet preko standardnog RJ-45 konektora. Tim modulom se povećavaju mogućnosti izrade programa jer tada Arduino nije samo uređaj koji radi u nekom lokalnom sustavu nego može uzimati i slati informacije na Internet tako da ih korisnici mogu dobivati i obrađivati na bilo kojem mjestu u samo nekoliko klikova miša! Baziran je na *Wiznetovom* W5100 čipu. Ima na sebi i uređaj za micro SD karticu no treba biti pažljiv jer se ne mogu koristiti SD kartica i *Ethernet* u isto vrijeme jer koriste isti SPI (eng. *Serial Peripheral Interface*) priključak na pinovima 10, 11, 12 i 13. Pin 4 se koristi za biranje SD kartice, a pin 10 za biranje *Wiznet* W5100 čipa, ti pinovi se obavezno moraju postaviti kao *OUTPUT*! Da bi se modul koristio, u C programu je potrebno uključiti njegove knjižnice, ethernet.h, spi.h i sd.h pomoću standardne #include naredbe.



Slika 7 - Izgled Arduino Ethernet Shielda

2.2.4. SD kartica

SD kartica [6] je trajna memorijska kartica intenzivno korištena u prenosivim uređajima poput mobilnih telefona, GPS uređaja, digitalnih kamera i sl. Pripada obitelji SSS (eng. *Solid State Storage*) medija. To znači da prihvaćanje digitalnih informacija vrše samo čiste elektroničke komponente za razliku od hard diskova na računalima koji koriste mehaničku vrtnju medija prekrivenog magnetskim slojem. SD kartice mogu imati 4 tipa sučelja priključka, a to su :

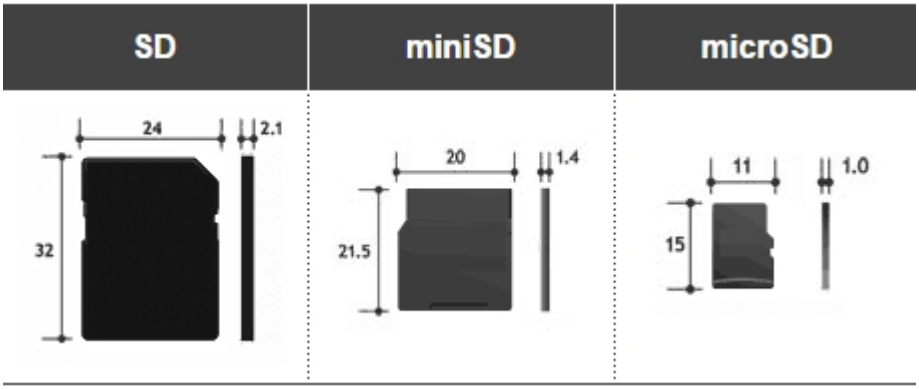
- *Normal speed* čija je maksimalna brzina prijenosa podataka 12.5MB/s
- *High speed* čija je maksimalna brzina prijenosa podataka 25MB/s
- UHS-1 čija je maksimalna brzina prijenosa podataka 50MB/s ili 104MB/s ovisno o tipu kartice
- UHS-2 čija je maksimalna brzina prijenosa podataka 156MB/s ili 312MB/s ovisno o tipu kartice

Minimalnu brzinu pisanja na SD karticu određuje njena klasa koja iznosi 2,4,6,8,10. Za UHS tip priključka postoje klase 1 i 3. Veći broj klase iznosi veću minimalnu brzinu pisanja na karticu.

SD kartice uključuju 4 obitelji kartica, a to su:

- SDSC (eng. *Secure Digital Standard Capacity*) ili samo SD koje imaju kapacitet memorije do 2 GB

- SDHC (eng. *Secure Digital High Capacity*) koje imaju kapacitet memorije u rasponu od 2 – 32GB
- SDXC (eng. *Secure Digital Expanded Capacity*) koje imaju kapacitet memorije u rasponu od 32GB – 2TB



Slika 8 - Prikaz veličina SD kartica [5]

3. Korišteni softver

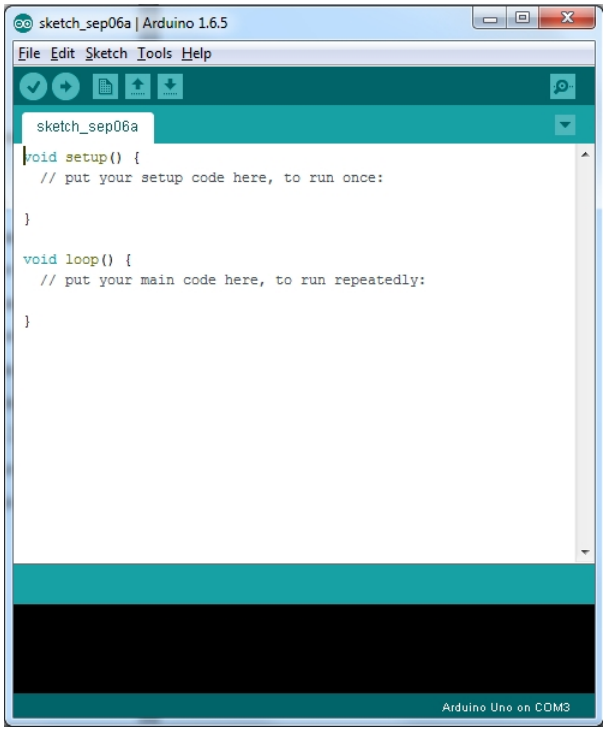
Za realiziranje projekta potrebni su bili sljedeći programi:

- Arduino programsko sučelje
- WAMP paket (web poslužitelj, Apache + mySQL poslužitelj baze podataka)
- Baza podataka

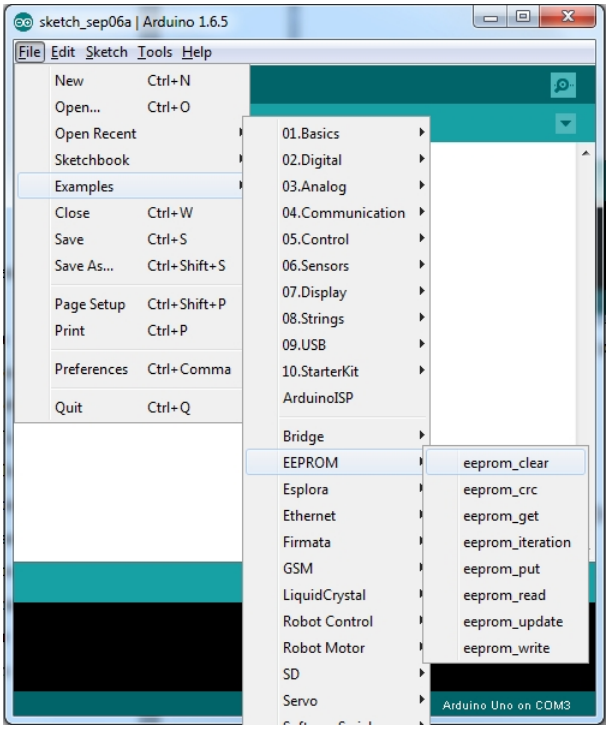
3.1. Arduino programsko sučelje

Arduino razvojno sučelje, uključujući razvojno okruženje IDE (eng. *Integrated Development Environment*) potpuno je besplatno te se može skinuti na službenim stranicama Arduina [1]. Programira se u programskom jeziku C/C++.

Izgled sučelja prikazan je na slici 9. Najvažniji dijelovi sučelja za programera su kompajliranje, slanje programa na Arduino te *Serial monitor*. Kompajliranje je proces prevođenja programa koji je napisan u jeziku koji je razumljiv čovjeku u strojni jezik. Programi za Arduino nazivaju se skice (eng. *Sketch*) i sastoje se od 2 dijela: *Setup* i *loop*. U *Setup* dijelu koda programiramo naredbe koje želimo da se prve izvrše te se izvršavaju samo jednom. To su većinom naredbe za definiranje pinova koji će se koristiti u daljnjem dijelu programa. U *loop* dio koda programiramo događaje koje želimo da se ponavljaju cijelo vrijeme (npr. čitanje stanja vage). *Serial monitor* je vrlo koristan jer nam vraća korisne informacije o izvršavanju programa. Sa programom Arduino.ide dobiju se i gotovi primjeri programa kako bi ljudi što lakše mogli započeti učenje programiranja na tom mikrokontroleru. Izbor gotovih primjera programa prikazan je slikom 10. Vrlo su korisni primjeri programa za pojedine knjižnice jer se upoznajemo sa sintaksom gotovih naredbi koje čine programiranje Arduina jednostavnijim od klasičnog programiranja u standardnim programskim jezicima.



Slika 9 - Izgled programskog sučelja Arduino



Slika 10 - Biranje gotovih primjera programa

3.2. Web poslužitelj

Web poslužitelj korišten u ovom radu je *Apache* koji je za potrebe testnog okruženja instaliran kao dio WAMP paketa kojeg se može preuzeti na sljedećoj web stranici: http://www.wampserver.com/. Nakon instalacije paketa automatski se stvori u njegovom direktoriju mapa pod nazivom „www“. U tu mapu dodajemo PHP skripte koje poslužitelj nakon pokretanja izvršava. Komunikacijski protokol koji se koristi za komunikaciju između web poslužitelja i Arduina zove se HTTP (eng. *Hyper Text Transfer Protocol*). Za web poslužitelj potrebno je napraviti i bazu podataka u koju će se spremati podaci dobiveni iz izrađenog sklopa. Nju ćemo napraviti u MySQL-u.

3.2.1. HTTP

HTTP (eng. *Hyper Text Transfer Protocol*) je protokol koji radi na aplikacijskom sloju TCP/IP modela. Najčešće je korišteni protokol pri prijenosu informacija na Internetu. On je baziran na upit/odgovor metodi za komunikaciju između poslužitelja i klijenta. [8] HTTP klijent poput web preglednika u većini slučajeva inicira prijenos informacija nakon što uspostavi vezu sa web poslužiteljem na određenom portu.

Poslužitelj konstantno osluškuje upite na određenom portu (obično port 80), čekajući da klijent pošalje niz znakova (*string*), kao što je "GET / HTTP/1.1" kojim će zahtjevati uspostavljanje komunikacije te nakon toga i tekstualnu poruku koja sadrži nekoliko slovnih nizova (zaglavlje) koji određuju aspekte zahtjeva te paket neobaveznih podataka. Upit klijenta će rezultirati slanje statusa poslužitelja, te sadržaj. Odmah po ispunjenju zahtjeva klijenta, poslužitelj će prekinuti komunikaciju.

3.2.2. Baze podataka

Baza podataka je organizirani skup podataka koji se može lako čitati i prikazivati. Podaci se u bazama podataka najčešće spremaju u tablice. Podaci se tada iz tablica uzimaju ovisno o potrebama korisnika. Korisnik u svakom trenutku mora imati na raspolaganju brzu bazu podataka, u suprotnom ta baza nakon dužeg perioda korištenja postane neuporabljiva zbog prevelikog vremena čekanja od zadavanja zahtjeva za podatak i njegovog dobivanja.

Postoji više različitih baza podataka koje možemo koristiti. U ovom projektu je odabran MySQL jer je jednostavna i besplatna inačica baze.

MySQL [10] je čest izbor baze za projekte otvorenog koda te se distribuira kao sastavni dio poslužiteljskih Linux distribucija, no također postoje inačice i za ostale operacijske sustave poput Mac OS-a, Windowse itd. MySQL baza je slobodna za većinu uporaba. Ranije u svom razvoju, MySQL baza podataka suočila se s raznim protivnicima MySQL sustava organiziranja podataka jer su joj nedostajale neke osnovne funkcije definirane SQL standardom. Naime, MySQL baza je optimizirana kako bi bila brza nauštrb funkcionalnosti. Nasuprot tome, vrlo je stabilna i ima dobro dokumentirane module i ekstenzije te podršku od brojnih programskih jezika: PHP, Java , Perl, Python. MySQL baze su relacijskog tipa, koji se pokazao kao najbolji način skladištenja i pretraživanja velikih količina podataka i u suštini predstavljaju osnovu svakog informacijskog sustava, tj. temelj svakog poslovnog subjekta koji svoje poslovanje bazira na dostupnosti kvalitetnih i brzih informacija.

4. Praktični rad

Praktični rad u ovom završnom radu bio je realiziranje digitalne vage pomoću Arduino platforme, zapisivanje izmjerene vrijednosti na SD karticu u obliku .csv datoteke te zapisivanje vrijednosti mjerenja u bazu podataka. Podatke spremljene u bazu podataka prikazujemo u obliku web stranice koja je izvedena dinamički i postavljena na web poslužitelj. Dinamički je izvedena PHP programskim jezikom te čita podatke iz baze podataka te ih vizualizira. Bilo je potrebno napraviti i aplikaciju u Excelu koja će čitati .csv datoteku te na temelju nje grafički prikazivati rezultate sa mogućnošću sortiranja. Također je trebalo napraviti da Arduino dijeli sadržaj svoje SD kartice te se njemu može pristupati preko web stranice našeg izrađenog poslužitelja te se mogu podaci čitati i vizualizirati iz .csv datoteke. To se postiže tako da Arduino radi kao poslužitelj te čeka GET upit. Nakon što dobije GET upit, Arduino vraća odgovor u kojem se nalazi lista svih datoteka koje se nalaze na SD kartici. Nakon što dobijemo listu svih datoteka, možemo odabrati datoteku kojoj želimo vidjeti sadržaj. Nakon što dobijemo sadržaj datoteke, iz njega crtamo graf koji nam vizualno predočuje promjenu vrijednosti vaganja.

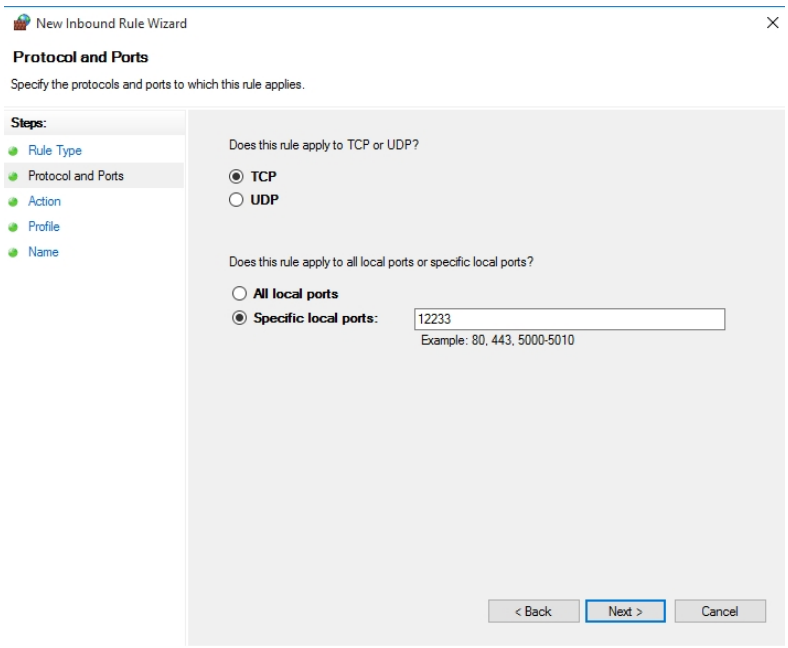
Oprema koja je bila potrebna u ovom radu opisana je u prethodnim poglavljima, a to su: Arduino uređaj, senzor težine, HX711 analogno digitalni pretvornik sa pojačalom, Arduino *Ethernet Shield* te micro SD kartica.

Cijeli kod za Arduino te web dio nalazi se u prilogu.

4.1. Poslužitelj i web stranica

Na poslužitelj baze podataka potrebno je pohraniti vrijednosti mjerenja digitalne vage, a web/HTTP poslužitelj mora omogućiti prikaz i jednostavnu vizualizaciju podataka. Poslužitelj je napravljen pomoću *Apache* poslužitelja. Za pohranjivanje podataka koristimo MySQL bazu podataka. Potrebna je vrlo jednostavna baza koja ima 1 tablicu sa 3 stupca. To su ID, datum i vrijeme mjerenja te vrijednost mjerenja izražena u kilogramima. Korištene tehnologije su: PHP, jQuerry te HTML.

Potrebno je definirati *port* koji poslužitelj osluškuje. Odabran je *port* 12233 umjesto 80 zbog toga jer se želi izbjeći mogući konflikt sa ostalim programima. Da bi se mogli slati podaci na taj port, potrebno ga je otvoriti u Windows postavkama vatrozida. Najprije odaberemo „*Inbound rules*“ te pritisnemo „*New rule*“ kako bi napravili novo pravilo. U tom pravilu upišemo *port* koji želimo otvoriti te da se to odnosi na TCP protokol.



Slika 11 - Otvaranje porta

Nakon što otvorimo *port* potrebno je napraviti bazu podataka u MySQL-u.

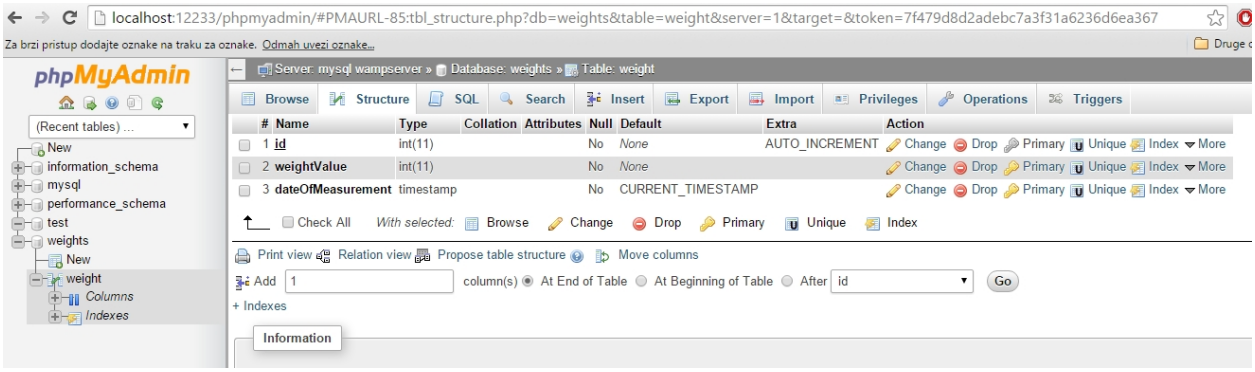
Prvo je potrebno izvršiti identifikaciju entiteta i atributa naše baze. Entitet je u našem slučaju Težina (*Weight*), a njegovi atributi su: Id, vrijeme vaganje (dateOfMeasurement) i vrijednost vaganja (weightValue). Nakon toga potrebno je napraviti rječnik podataka.

Naziv atributa	Tip	Opis
Id	Cijeli broj (*INT*)	Redni broj mjerenja
dateOfMeasurement	Datum i vrijeme (*TIMESTAMP*)	Datum i vrijeme u trenutku mjerenja
weightValue	Cijeli broj (*INT*)	Vrijednost vaganja

Tablica 3 - Rječnik podataka za bazu podataka

Nakon toga potrebno je odrediti primarni ključ, a to će u ovom slučaju biti id iz tog razloga jer je id jedini podatak koji će zasigurno biti različit za svako mjerenje i time može jednoznačno određivati primjerak entiteta.

Na kraju izrađujemo bazu podataka korištenjem alata phpMyAdmin koji je dio instaliranog WAMP paketa. Alat se pokreće upisivanjem sljedeće adrese u web preglednik: http://localhost:12233/phpmyadmin/. Otvori se okruženje u kojem izrađujemo bazu podataka prema našim zahtjevima koje smo obradili.



Slika 12 - Struktura kreirane tablice u bazi podataka

Nakon što smo kreirali bazu podataka potrebno je napraviti PHP dio za spajanje na bazu podataka.

Da bi se spojili na bazu podataka koristimo PDO PHP *driver* za pristup bazi podataka. Njemu je potrebno definirati *driver* baze podataka, ime tablice, IP adresu koja je u našem slučaju „*localhost*“, korisničko ime, password te *port* koji je za MySQL bazu podataka 3306. Nakon tih definiranih parametara bazi pristupamo sljedećom naredbom:

```
$db = new PDO($dns, $dbuser, $dbpass ,
          array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES \"UTF8\"",
          PDO::ATTR_PERSISTENT => false));
```

Kod za spajanje na bazu podataka je sljedeći:

```
<?php
	$dbdriver = "mysql";
	$dbname = "weights";

	$dbhost = "localhost";
	$dbuser = "root";
	$dbpass = "";
	$dbport = "3306";

	$dns = $dbdriver . ":host=" . $dbhost . ";dbname=" . $dbname . ";port=" .
		$dbport;

	$try = 0;
	try{
		$db = new PDO($dns, $dbuser, $dbpass ,
			array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES
			\"UTF8\"", PDO::ATTR_PERSISTENT => false));
	}
	catch(Exception $e){
		echo "fail";

	}
?>
```

Nakon što napravimo PHP dio programa za spajanje na bazu podataka potrebno je napraviti PHP kod za dohvaćanje podataka iz baze:

```
<?php

	include ("ConnectToDatabase.php");

	if (isset($_GET["date"])){
		$queryDate = $_GET["date"];
		$date1 = new DateTime($queryDate . ' 00:00:00.00000');
		$date2 = new DateTime($queryDate . ' 23:59:59.99999');
	}

	$queryString = "SELECT weightValue, dateOfMeasurement FROM weight ";
	if (isset($queryDate)){
		$queryString .= " WHERE dateOfMeasurement BETWEEN (?) AND (?)";
	}
	$queryString .= " ORDER BY dateOfMeasurement DESC";

	$data = array();
	$query = $db->prepare($queryString);
	if (isset($queryDate))
		$query->execute(array($queryDate . ' 00:00:00.00000', $queryDate .
				' 23:59:59.99999'));
	else
		$query->execute();
    $result = $query->fetchAll();

	foreach ($result as $singleData){
		array_push($data, array("date" =>
				$singleData["dateOfMeasurement"], "weight" =>
				$singleData["weightValue"]));
	}

	echo json_encode($data);
?>
```

Za dohvaćanje podataka iz baze podataka potrebno se prvo spojiti na bazu podataka PHP skriptom koja je prethodno objašnjena. Nakon učitavanja te skripte šalje se upit prema bazi podataka `$query` naredbom. Šaljemo SQL upit koji glasi: „`SELECT weightValue, dateOfMeasurement FROM weight ORDER BY dateOfMeasurement DESC`“. Tim upitom dobivamo iz baze podataka vrijednost mjerenja te datum i vrijeme mjerenja sortirano po datumu od najnovijeg prema najstarijem. Ako odaberemo određeni parametar datuma tada se šalje i „`WHERE dateOfMeasurement BETWEEN XX:XX:XXXX:00:00:00 AND XX:XX:XXXX:23:59:59`“ gdje XX:XX:XXX predstavlja datum.

Nakon dohvata podataka rezultat šaljemo red po red te datum i vrijeme te vrijednost mjerenja postavljamo kao pojedinačne vrijednosti kako bi se kasnije mogle uzimati kao zasebne vrijednost te vršiti operacije nad njima.

Na kraju podatke enkodiramo JSON formatom zapisa. JSON format je format koji koristi tekst koji je razumljiv čovjeku za prijenos objekta podataka koji se sastoje od atributa parova.

Kod za pisanje u bazu podataka vrši se tako da poslužitelj čeka GET zahtjev od Arduina te kad ga primi tada pročita vrijednost mjerenja koju je Arduino poslao te ju zapiše u bazu podataka SQL naredbom „`INSERT INTO weight (weightValue) VALUES ((?))`“ U kojoj umjesto simbola „?“ napiše vrijednost koju je Arduino poslao.

Kod za pisanje u bazu podataka je sljedeći:

```
<?php

	include ("ConnectToDatabase.php");
	$data = json_decode($_GET["weight"], true);
	$statement = $db->prepare("INSERT INTO weight (weightValue) VALUES
			((?))");
	$statement->execute(array($data));

?>
```

Na web stranici stvaramo tablicu pomoću HTML-a koja ima 2 stupca imenovana *Date of Measurement* te *Weight value*. U retke zapisujemo datum te vrijednost mjerenja u predviđeni stupac za to:

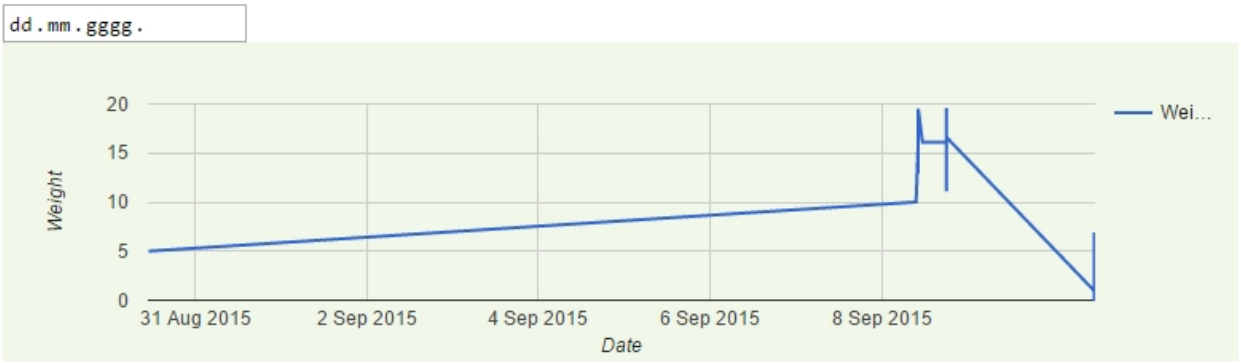
```
   <table>
	<thead>
		<tr>
			<th>Date of Measurement</th>
			<th>Weight value</th>
		</tr>
	</thead>
	<tbody>
		<?php foreach ($result as $singleData) {?>
			<tr>
				<td>
					<span><?php echo
						$singleData["dateOfMeasurement"]
						?></span>
				</td>
				<td>
					<span><?php echo
						intval($singleData["weightValue"])
						/10 ?> kilograms</span>
				</td>
			</tr>
		<?php } ?>
	</tbody>
   </table>
```

Graf je izrađen pomoću *Google Chart* API. API [11] (eng. A*pplication Programming Interface*) Aplikacijsko programsko sučelje za programiranje aplikacija je skup određenih pravila i specifikacija koje programeri slijede tako da se mogu služiti uslugama ili resursima operacijskog sustava ili nekog drugog složenog programa. API koji je korišten za ovaj graf može se skinuti na sljedećem linku: https://jsfiddle.net/api/post/library/pure/. *Google Charts* API je API kojeg pruža Google pomoću kojeg zadani skup podataka možemo prikazati u grafu. Prvo je potrebno odabrati koji tip grafa želimo, unijeti podatke u varijablu koju daje API, a zatim Google API treba ID HTML elementa kako bi znao unutar kojeg elementa je potrebno iscrtati graf.

Google charts dio koda:

```
<script type="text/javascript" src="jquery-1.11.3.min.js"></script>
<script type="text/javascript" src="https://www.google.com/jsapi"></script>
<script type="text/javascript">
	google.load('visualization', '1', {packages: ['corechart',
				'line']});
	google.setOnLoadCallback(initializeTable);
```

```
        function initializeTable(){
                var data = new google.visualization.DataTable();
                data.addColumn('datetime', 'X');
                data.addColumn('number', 'Weight');
                var url = "http://localhost:12233/weights/getWeights.php";
                if ($('#date').val()){
                        url += '?date=' + $('#date').val();
                }
                $.get(url, function(weights){
                        weights = JSON.parse(weights);
                        $.each(weights, function(index, singleWeight){
                                data.addRows([[new Date(singleWeight["date"]),
                                        parseInt(singleWeight["weight"])/10]]);
                        });
                        var options = {
                                hAxis: {
                                        title: 'Date'
                                },
                                vAxis: {
                                        title: 'Weight'
                                },
                                backgroundColor: '#f1f8e9'
                        }
                        var chart = new google.visualization.LineChart
                                        (document.getElementById('chart_div'));
                        chart.draw(data, options);
                });
        }
</script>
```



Date of Measurement	Weight value
2015-09-10 11:31:48	6.1 kilograms
2015-09-10 11:31:40	6.1 kilograms
2015-09-10 11:31:32	6.7 kilograms
2015-09-10 11:31:23	6.9 kilograms
2015-09-10 11:31:15	4.2 kilograms
2015-09-10 11:31:06	0 kilograms
2015-09-10 11:30:58	0 kilograms
2015-09-10 11:30:50	0 kilograms
2015-09-10 11:29:35	1 kilograms

Slika 13 - Izgled web stranice

4.1.1. Čitanje podataka sa Arduinove SD kartice

Za ovaj dio potrebno je Arduino postaviti kao Poslužitelj kako bi nam mogao dijeliti sadržaj svoje SD kartice. Šaljemo GET zahtjev na Arduino poslužitelj te čekamo odgovor. Kad primimo odgovor, zamijenimo URL sa našim:

```
<?php
	$curl = curl_init();
	curl_setopt_array($curl, array(
	 CURLOPT_RETURNTRANSFER => 1,
	 CURLOPT_URL => 'http://192.168.1.16'
	));
	$resp = curl_exec($curl);
	curl_close($curl);

	echo str_replace("href=\"","class=\"arduinoFileLink\"
				href=\"getFileFromArduino.php?filename=",$resp);
?>
```

U sljedećem dijelu koda šaljemo drugi GET zahtjev na Arduino poslužitelj kojim biramo datoteku koju želimo otvoriti. Arduino poslužitelj nam vraća odgovor sa sadržajem datoteke koji prosljeđujemo na našu web stranicu:

```
<?php
	$filename = $_GET["filename"];
	$curl = curl_init();
	curl_setopt_array($curl, array(
	 CURLOPT_RETURNTRANSFER => 1,
	 CURLOPT_URL => 'http://192.168.1.16/' . $filename));
	$resp = curl_exec($curl);
	curl_close($curl);
	echo $resp;
?>
```

U sljedećem dijelu koda stvaramo gumb. Pritiskom gumba izvrši se dio koda za biranje datoteke koju želimo otvoriti na Arduinu. Nakon otvaranja liste ponuđenih datoteka na SD kartici biramo koju datoteku želimo otvoriti. Kada odaberemo datoteku, izvrši se kod za ispis sadržaja datoteke te se stvara tablica koja ima 2 stupca, datum i vrijeme te vrijednost mjerenja.

Graf se radi po istom API-u kao i u poglavlju 4.1 sa razlikom što sad dobiva podatke iz GET odgovora koji šalje Arduino poslužitelj:

```
<!DOCTYPE html>
<html>
<head>
    <title>Weight Tracker</title>

</head>
<body>
	<button type="button" id="getFromArduino">Check what Arduino has to
			offer!</button>
	<div id="arduinoResponse"></div>
	<div id="chart_div">
	</div>
    <table id="weightTable">
		<thead>
			<tr>
				<th>Date of Measurement</th>
```

```
                        <th>Weight value</th>
                    </tr>
                </thead>
                <tbody>
                </tbody>
        </table>
        <script type="text/javascript" src="jquery-1.11.3.min.js"></script>
        <script type="text/javascript"
                     src="https://www.google.com/jsapi"></script>
        <script type="text/javascript">
                google.load('visualization', '1', {packages: ['corechart',
                        'line']});

                function initializeTable(url){
                        var data = new google.visualization.DataTable();
                        data.addColumn('datetime', 'X');
                        data.addColumn('number', 'Weight');

                        $.get(url, function(weights){
                                weights = weights.replace(/ /g, '');
                                weights = weights.split('\n');
                                $('#weightTable > tbody').html('');
                                $.each(weights, function(index, singleWeight){
                                        if (!singleWeight) return true;
                                        var row = singleWeight.split(';');
                                        var dateCol = row[0].split(':');
                                        var timeCol = row[1].split(':');
                                        var weight = parseFloat(row[2].replace(/,/g,
                                                        '.'));

                                        var date = new Date(dateCol[2], dateCol[1],
                                                        dateCol[0], timeCol[0], timeCol[1],
                                                        timeCol[2]);

                                        data.addRows([[date, weight]]);
                                        $('#weightTable > tbody:last-
                                                child').append('<tr><td>' +
                                                date.getDay() + '.' +
                                                (date.getMonth()+1).toString() + '.'
                                                + date.getFullYear().toString() +
                                                '.</td><td>' + weight +
                                                '</td></tr>');
                                });
                                var options = {
                                        hAxis: {
                                                title: 'Date'
                                        },
                                        vAxis: {
                                                title: 'Weight'
                                        },
                                        backgroundColor: '#f1f8e9'
                                }
                                var chart = new
                                                google.visualization.LineChart
                                                (document.getElementById('chart_div'));
                                chart.draw(data, options);
                        });
                }
        </script>
        <script type="text/javascript">
                $("#getFromArduino").click(function(){
                        $.get('checkArduino.php', function(response){
                                $('#arduinoResponse').html(response);

                                $('.arduinoFileLink').click(function(event){
                                        var element = $(event.target);
                                        var link = element.attr('href');
                                        initializeTable(link);
                                        return false;
                                });
```

```
                    })
                });
        </script>
</body>
</html>
```

Slika 14 - Izgled web stranice za čitanje podataka iz .csv datoteke

4.2. Arduino dio

4.2.1. Spajanje komponenti

Prvi korak u izradi fizičkog dijela rada bio je proučiti komponente koje se koriste te na temelju toga spojiti ih na Arduino platformu kako bi se mogao početi pisati upravljački program.

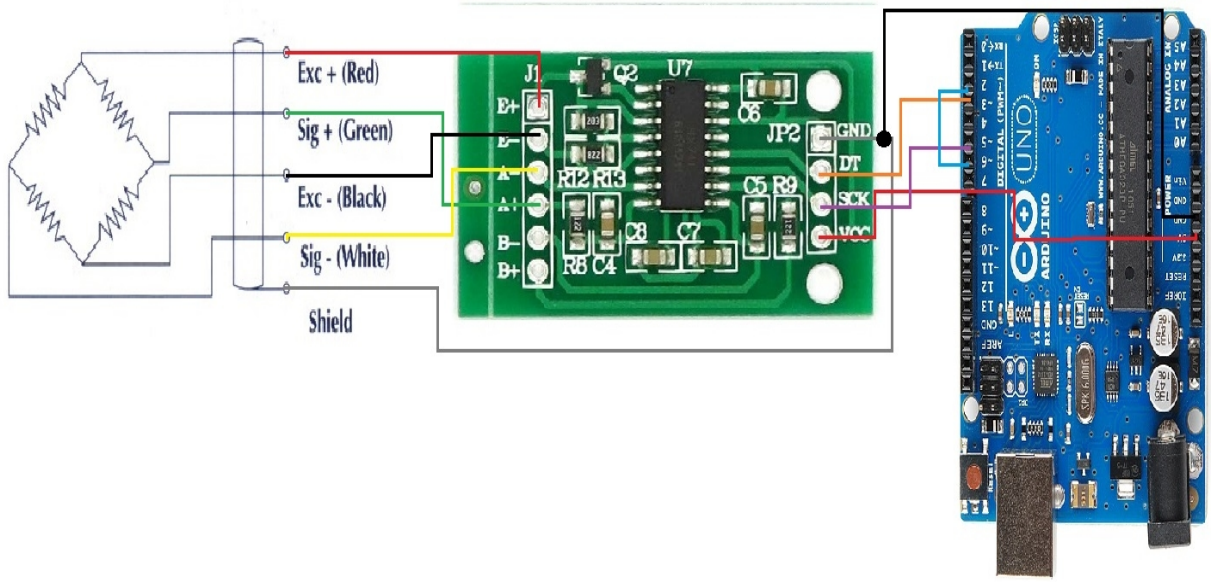
Najprije je potrebno spojiti *Ethernet Shield* na osnovnu Arduino Uno pločicu kako bi se dobio pristup Internetu preko Ethernet kabla te pristup SD kartici preko SPI priključka. *Ethernet Shield* koristi pinove 10,11,12,14 te pin 4. Pin 10 koristi se za omogućavanje komunikacije preko Interneta, a pin 4 za omogućavanje komunikacije sa SD karticom.

Nakon toga bilo je potrebno realizirati spoj za vrijeme na uređaju. Postoje moduli za čuvanje vremena koji u sebi imaju bateriju, ni iz razloga što nam modul nije bio dostupan bilo je potrebno koristiti drugo rješenje. Bile su dvije mogućnosti, koristiti Arduino naredbu `millis()` koja vraća broj milisekundi od kad je uređaj upaljen. Glavni problem te naredbe je taj da će se registar u kojem je pohranjena ta vrijednost u nekom trenutku prebaciti na nulu. Uz to postoji i neko malo kašnjenje pa ni ta milisekunda koja se dobije iz naredbe nije točno jedna milisekunda nego varira za neki mali postotak. Odlučili smo koristiti drugo rješenje koje je preciznije, a to je korištenje pulsno širinske modulacije. Pin 6 je bilo potrebno spojiti na digitalni pin 2. Pin 6 ima

mogućnost slanja pulsno širinskih signala frekvencije 974Hz. Pin 6 podešen je kao izlaz dok je pin 2 ulaz.

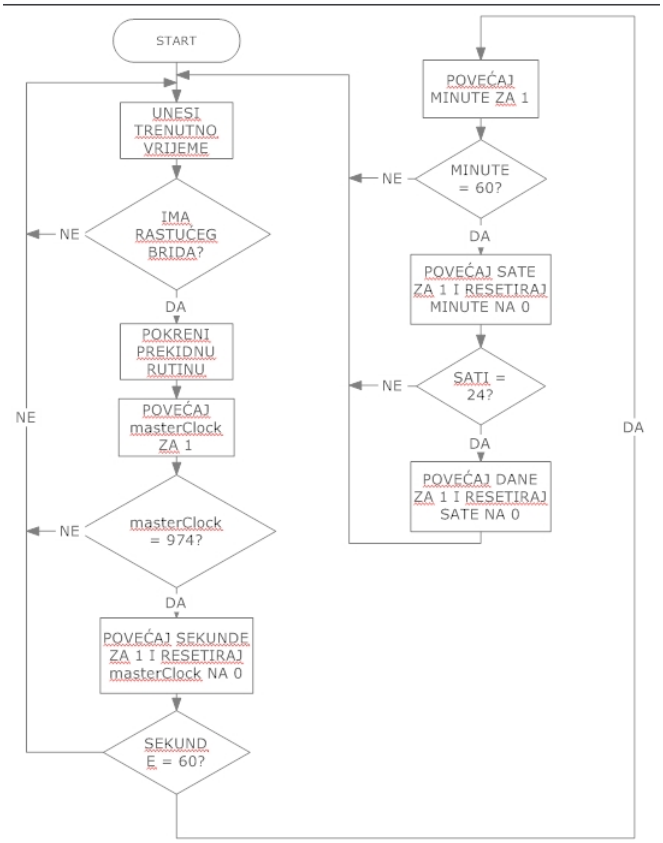
Nakon potrebnog spoja za vrijeme, bilo je potrebno spojiti HX711 modul na Arduino. Potrebno je analogno digitalnom pretvorniku spojiti napon od 5 volti iz Arduina te spojiti DOUT i SCK pinove na digitalne pinove po izboru, u našem slučaju su to pin 3 i pin 5. Spoj je prikazan na slici 15.

Na kraju je potrebno spojiti senzor težine na HX711. To se postiže tako da se napajanje za senzor uzme iz pinova E+ i E- analogno digitalnog pretvornika, a izlaz senzora na kanal A zbog toga jer iz njega možemo dobiti najveće pojačanje koje iznosi 128 puta. *Shield* žica senzora spaja se na masu.



Slika 15 - Spoj komponenti uređaja

4.2.2. Programiranje vremena



Slika 16 - Blok dijagram za vrijeme

Za programiranje vremena kako je već spomenuto u prethodnom poglavlju koristimo pinove 2 i 6. Najprije trebamo kreirati varijable za sekunde, minute, sate, dane itd. te varijablu nazvanu *masterClock* koja će brojiti rastuće bridove. Glavna logika programa prikazana je blok dijagramom na slici 16. Pin 6 šalje impulse frekvencijom 974Hz što znači da će na pin 2 doći 974 pravokutnih bridova u jednoj sekundi. Za svaki rastući brid pokreće se prekidna rutina nazvana *clockCounter*. Ona je definirana u *setup* dijelu programa sljedećom naredbom:

```
attachInterrupt(clockInterrupt, clockCounter, RISING);
```

Zbog toga jer se prekidna rutina pokreće na svaki pozitivni brid, dodajemo + 1 u *masterClock* varijablu te vršimo provjeru preljeva zbog toga jer ne želimo da nam se preljev događa na vrijednosti (2^{31} -1) nego na 60 za minute i sekunde te na 24 za sate. Za dane nismo vršili provjeru preljeva jer ovaj sklop nije predviđen da radi više od mjesec dana bez resetiranja jer mu se s vremenom povećava pogreška zbog senzora težine. Kod za ovaj dio programa je sljedeći:

```
void clockCounter() {
  masterClock ++;
  if(masterClock == 974) {
```

```
      seconds ++;
      masterClock = 0;

      if(seconds==60){
       minutes++;
       seconds = 0;

        if(minutes==60){
          hours++;
          minutes = 0;
        }
        if(hours==24){
          days++;
          hours = 0;
        }
      }
    }
  return;
  }
```

4.2.3. Programiranje čitanja stanja vage

Za ovaj dio koda potrebno je prvo uključiti knjižnicu [7] za analogno digitalni pretvornik HX711. Knjižnice su zapravo grupe unaprijed definiranih funkcija zbog kojih nam je puno lakše pisati kod. Budući da to nije standardna knjižnica potrebno ju je skinuti na web stranici https//:github.com. Na toj web stranici ima mnogo korisnih knjižnica prilagođenih za module koji se koriste s Arduino pločicom, a nisu uključene u standardno izdanje programa. Da bi se nova knjižnica mogla koristiti jednostavno se sadržaj skinute .rar datoteke kopira u *library* direktorij Arduino mape na računalu. Uz knjižnicu dobije se primjer korištenja naredbi koje su uključene u nju te opis kako bi što lakše shvatili kako se koristi.

Najprije se trebaju definirati SCK i DOUT pinovi. To se radi naredbom:

```
HX711 scale(3, 5);
```

Prvi parametar predstavlja DOUT, a drugi SCK. Nakon definiranja pinova potrebno je u *setup* funkciji pozvati naredbe:

```
scale.setScale(14605.f);
scale.tare();
```

Prva linija predstavlja postavljanje koeficijenta vaganja. Broj u zagradi se dobije kalibracijom koju ćemo kasnije i objasniti. Druga linija koda postavlja pročitanu vrijednost kao nulu prilikom uključenja uređaja.

Naredba za dohvat podataka je sljedeća:

```
scale.get_units(10),1;
```

Ta naredba treba 2 parametra, prvi (10) znači da će napraviti 10 čitanja te uzeti njihov prosjek kao rezultat. Time se dobije precizniji rezultat te se uklone neke male oscilacije koje se javljaju u tako osjetljivom senzoru. Drugi parametar predstavlja broj decimala.

Kalibriranje vage se vrši na istom kodu kao i kad imamo kalibriranu vrijednost. Jedina je razlika što se naredba `scale.setScale` poziva bez parametara. Pokrene se Arduino te *Serial Monitor* te nakon nekoliko sekundi stavimo poznat teret na vagu. S obzirom da ovaj senzor može vagati do 300 kg preporučljivo je da se kalibrira na nekoj malo većoj težini npr. 80 kilograma. Brojevi koje dobivamo na *Serial Monitoru* predstavljaju čistu vrijednost vaganja. Taj broj podijelimo sa 80 kilograma koje smo stavili na vagu te dobijemo koeficijent koji je potrebno staviti u `scale.setScale();` naredbu (u našem slučaju je to 14605.f).

4.2.4. Programiranje SD kartice

Uloga SD kartice u ovom radu je ta da na sebi ima .csv datoteku koju ćemo čitati u Excel aplikaciji. Format zapisa je sljedeći: Dan:Mjesec:Godina;Sat:Minuta;Vrijednost mjerenja.

Prvo trebamo učitati knjižnice koje su nam potrebne za korištenje SD kartice, a to su `<SD.h>` te `<SPI.h>`. Nakon toga imenujemo primjer otvorene datoteke kao myFile. Jako je bitno da se pinovi 4 i 10 postave kao *OUTPUT* te da ih se postavi kao *HIGH* na početku programa. Ti pinovi služe za odabir da li želimo koristiti SD karticu ili čip za *Ethernet*. Kada su oba postavljena kao *HIGH* tada nije moguće koristiti ni jedan ni drugi. U programu ih stavljamo u *LOW* samo kada ih koristimo za slanje te nakon toga odmah postavimo u *HIGH* zbog toga da znamo da ni u jednom trenutku nisu i jedan i drugi omogućeni jer u tom slučaju ne bi bilo moguće ni slanje na SD karticu ni slanje na poslužitelj. To se postiže ovim naredbama:

```
pinMode(SS_SD_CARD, OUTPUT);
pinMode(SS_ETHERNET, OUTPUT);
digitalWrite(SS_SD_CARD, HIGH);
digitalWrite(SS_ETHERNET, HIGH);
```

Nakon toga provjeravamo *If* funkcijom da li je kartica učitana ili ne. Ako nije bitno nam je da nam ispiše na *Serial Monitor* da je nema kako bi znali u čemu je problem u programu.

Ako je sa programom sve u redu vrijeme je za pisanje u datoteku. Čitanje stanja vage kao i njezin zapis događa se svakih 5 sekundi. Pin za SD karticu definiramo kao *LOW* kako bi omogućili pristup pisanju. Definiranom *myFile* dodajemo lokaciju datoteke u koju želimo pisati te parametar FILE_WRITE kojim definiramo da ćemo pisati u datoteku, a ne čitati. Ako datoteka postoji, zapišemo datum i vrijeme te iznos vaganja. U suprotnom ispišemo u *Serial Monitor* da se dogodila greška pri otvaranju datoteke. Ta greška znači da na SD kartici nema datoteke sa tim

imenom koje mi tražimo. Nakon ispisa vratimo pin za SD karticu na stanje *HIGH*. Kod za pisanje na SD karticu je sljedeći:

```
digitalWrite(SS_SD_CARD, LOW);
  delay(10);
  myFile = SD.open("weights.csv", FILE_WRITE);
   if (myFile) {
     SDPrintTime();
     myFile.print(";");
     myFile.print(scale.get_units(10),1);
     myFile.close();
   }
   else {
       Serial.println("error opening weights.csv");
   }
  digitalWrite(SS_SD_CARD, HIGH);
```

4.2.5. Programiranje slanja GET zahtjeva

Cilj nam je uspostaviti vezu sa web poslužiteljem te mu poslati HTTP GET zahtjev kako bi zapisali vrijednost vaganja na njega. web poslužitelj objašnjen je u poglavlju 4.1. Slanje GET zahtjeva se obavlja odmah nakon zapisa na SD karticu.

Najprije učitamo knjižnicu <ethernet.h>. Odaberemo MAC (eng. *Media Access Control*) adresu po želji naredbom:

```
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
```

Slovne oznake se mogu odabrati proizvoljno ali treba pripaziti da MAC adresa bude jedinstvena u lokalnoj mreži u kojoj se nalazi.

Nakon MAC adrese moramo proizvoljno odabrati i IP adresu. Također treba voditi računa da ona bude jedinstvena u lokalnoj mreži. Zadnja adresa koju treba definirati jest IP adresa poslužitelja. Pozovemo naredbu kojom započinjemo korištenje ethernet modula sa MAC adresom i IP adresom koju smo odabrali:

```
Ethernet.begin(mac, ip);
```

Mogli smo ovu naredbu pozvati bez IP parametra te bi tada *Ethernet Shield* sam odabrao IP adresu pomoću DHCP (eng. *Dynamic Host Computer Protocol*) protokola. Taj dio bi povećao veličinu *sketch* datoteke tako da nam to nije potrebno. Prije spajanja na poslužitelj moramo omogućiti pin za korištenje *Wiznet* čipa za Internet komunikaciju.

Upitom

```
if (client.connect(hostip, 12233)) { }
```

ispitujemo da li se naš uređaj spojio na poslužitelj. Parametri su IP poslužitelja te njegov port na kojem sluša. Ako naš uređaj nije spojen na poslužitelj zapisujemo *error* poruku na *Serial Monitor*, a ako jest spojen izvršavamo GET upit koji izgleda ovako:

```
client.print("GET /weights/insert.php?weight=");
client.print(scale.get_units(10)*10,1);
client.print(" HTTP/1.1\r\n");
client.print("Host: 192.168.1.3\r\n");
client.print("User-Agent: arduino-ethernet\r\n");
client.print("Connection: close\r\n\r\n");
client.println();
```

Nakon izvršenja slanja prekidamo konekciju naredbom `client.stop();`

4.2.6. Programiranje Arduina da radi kao poslužitelj

Ovaj dio koda potreban nam je za web dio spomenut u poglavlju 4.1.1. Potrebno je Arduino podesiti da radi kao poslužitelj te da ispisuje sve datoteke koje se nalaze na njegovoj SD kartici. Ako korisnik pritisne link za određenu datoteku, poslužitelj mu ispiše sadržaj te datoteke. Koriste se HTTP GET zahtjevi.

Zbog premalo preostale memorije u prethodnom Arduino programu koji je objašnjen u prethodnim poglavljima bilo je potrebno napraviti potpuno novi program.

Za ovaj Arduino program potrebno je uključiti knjižnice <ethernet.h>, <SPI.h>, <SdFat.h> i <SdFatUtil.h>. Cijeli dio koda nalazi se u prilogu!

Prvo je potrebno promotriti funkciju *ListFiles*. Ona nam pomaže u zapisivanju liste datoteka koju vidimo na poslužitelju. Objekt *dir_t p* je „držač“ unosa u direktorij. On pohranjuje informaciju o svakom unosu u direktoriju.

Resetiramo *root* direktorij naredbom *rewind();*. Nakon toga čitamo datoteke u direktoriju datoteku po datoteku. Ignoriramo datoteke „.“ I „..“ koja predstavlja link za „*up*“ direktorij. U listi prikazujemo samo datoteke i poddatoteke! Nakon toga ispisujemo ime datoteke tako da čitamo svih 11 znakova (datoteke su zapisane u 8.3 formatu) i zanemarujemo razmake. Zapisujemo točku između prvih 8 znakova i zadnjih 3. Nakon svakog imena datoteke dodajemo znak „/“ kako bi naznačili da je to datoteka. Na kraju nakon svake datoteke odlazimo u novi red.

Funkcija *ListFiles*:

```
void ListFiles(EthernetClient client, uint8_t flags) {
  dir_t p;
  root.rewind();
  client.println("<ul>");
  while (root.readDir(&p) > 0) {
    if (p.name[0] == DIR_NAME_FREE) break;
    // preskoči izbrisane zapise i zapise za . ili ..
    if (p.name[0] == DIR_NAME_DELETED || p.name[0] == '.') continue;
    // ispisuj samo poddirektorije i datoteke
    if (!DIR_IS_FILE_OR_SUBDIR(&p)) continue;
```

```
      client.print("<li><a href=\"");
      for (uint8_t i = 0; i < 11; i++) {
        if (p.name[i] == ' ') continue;
        if (i == 8) {
          client.print('.');
        }
        client.print((char)p.name[i]);
      }
      client.print("\">");
      // ispiši ime datoteke sa mogućim popunjavanjem praznina
      for (uint8_t i = 0; i < 11; i++) {
        if (p.name[i] == ' ') continue;
        if (i == 8) {
          client.print('.');
        }
        client.print((char)p.name[i]);
      }
      client.print("</a>");
      if (DIR_IS_SUBDIR(&p)) {
        client.print('/');
      }
      // ispiši datum/vrijeme modificiranja datoteke ako je zatraženo
      if (flags & LS_DATE) {
         root.printFatDate(p.lastWriteDate);
         client.print(' ');
         root.printFatTime(p.lastWriteTime);
      }
      // ispiši veličinu ako je zatraženo
      if (!DIR_IS_SUBDIR(&p) && (flags & LS_SIZE)) {
        client.print(' ');
        client.print(p.fileSize);
      }
      client.println("</li>");
    }
    client.println("</ul>");
  }
```

U sljedećem dijelu koda čekamo klijenta naredbom `server.available();`. Čitamo klijentov zahtjev u međuspremnik *clientline* tako dugo do dok ne dobijemo simbol za novi red (\n ili \r). To nam znači da smo pročitali cijelu liniju teksta. Da „završimo“ *string* dodajemo „0“ na kraju naredbom `strstr(clientline, " HTTP"))[0] = 0;`. Nakon toga koristimo *strstr* naredbu koja će tražiti substringove. Ako dobijemo GET zahtjev „GET /HTTP“ za *root* direktorij, ispisujemo listu datoteka na SD kartici pomoću prethodno opisane funkcije.

Ako nemamo razmaka nakon „GET /“ to znači da trebamo podijeliti sadržaj datoteke koju je odabrao korisnik u web browseru. Pokušamo otvoriti odabranu datoteku te ako uspijemo ispišemo njen sadržaj, a ako ne uspijemo, vraćamo *error* „404 – *FILE NOT FOUND*“. Kod za ovaj dio programa je sljedeći:

```
  void loop()
  {
    char clientline[BUFSIZ];
    int index = 0;

    EthernetClient client = server.available();
    if (client) {
      // http zahtjev završava praznim redom
      boolean current_line_is_blank = true;

      // resetiraj ulazni međuspremnik
      index = 0;
```

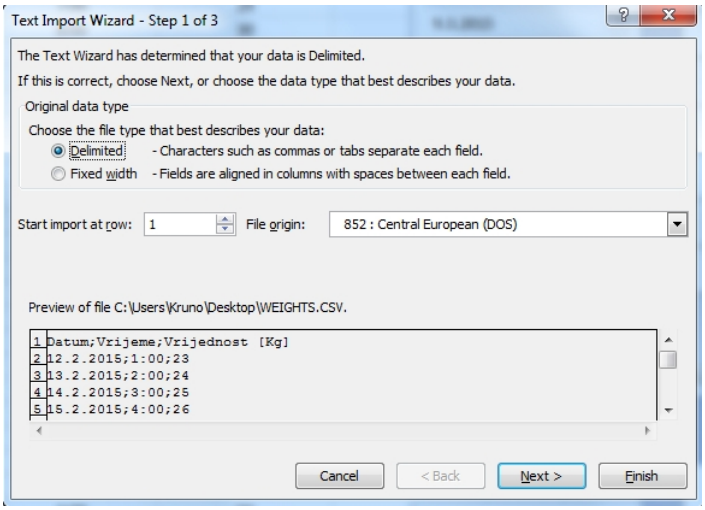
```
      while (client.connected()) {
        if (client.available()) {
          char c = client.read();

          //ako nije nova linija, dodaj znak u međuspremnik
          if (c != '\n' && c != '\r') {
            clientline[index] = c;
            index++;
            // ako imamo previše znakova u međuspremniku, izbaci jednog
            if (index >= BUFSIZ)
              index = BUFSIZ -1;

            // nastavi čitati podatke
            continue;
          }

          // gledaj da li je poslan \n ili \r što znači da je string gotov
          clientline[index] = 0;

          // ispiši što je klijent poslao
          Serial.println(clientline);

          if (strstr(clientline, "GET / ") != 0) {
            // pošalji standardno zaglavlje HTTTP odgovora
            client.println("HTTP/1.1 200 OK");
            client.println("Content-Type: text/html");
            client.println();

            // ispiši sve fileove
            client.println("<h2>Files:</h2>");
            ListFiles(client, LS_SIZE);
          } else if (strstr(clientline, "GET /") != 0) {

            char *filename;

            filename = clientline + 5;
            (strstr(clientline, " HTTP"))[0] = 0;

            // ispiši datoteku koja je odabrana
            Serial.println(filename);

            if (! file.open(&root, filename, O_READ)) {
              client.println("HTTP/1.1 404 Not Found");
              client.println("Content-Type: text/html");
              client.println();
              client.println("<h2>File Not Found!</h2>");
              break;
            }

            Serial.println("Opened!");

            client.println("HTTP/1.1 200 OK");
            client.println("Content-Type: text/plain");
            client.println();

            int16_t c;
            while ((c = file.read()) > 0) {
                client.print((char)c);
            }
            file.close();
          } else {
            // Sve ostalo je error 404 - file not found
            client.println("HTTP/1.1 404 Not Found");
            client.println("Content-Type: text/html");
            client.println();
            client.println("<h2>File Not Found!</h2>");
          }
          break;
        }
```

```
        }
        delay(1);
        client.stop();
      }
    }
```

4.3. Excel aplikacija

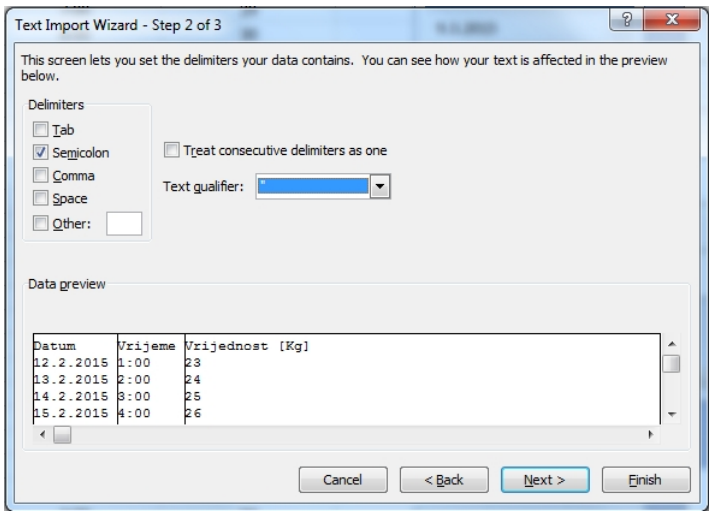
Excel aplikacija nam je potrebna kako bi dobili statističku analizu podataka zapisanih na SD kartici u .csv datoteci. Mogu se raditi razne analize koje su korisniku potrebne poput razlike između težine na početku dana i na kraju dana, na početku sata i na kraju sata itd. U ovom radu smo napravili graf koji se može sortirati po želji kako bi se i grafički mogle vidjeti promjene jer je korisniku puno lakše vizualno analizirati te podatke.

Unos podataka se vrši automatski uvozom podataka iz csv datoteke u stupce i retke tablice. Imamo 3 stupca koji predstavljaju datum mjerenja, vrijeme mjerenja i težinu predmeta na vagi. Da bi se definiralo iz koje će se datoteke ti podaci uvoziti te na koji način koristimo *Text Import Wizard*.



Slika 17 - Prvi korak u korištenju Text Import Wizarda

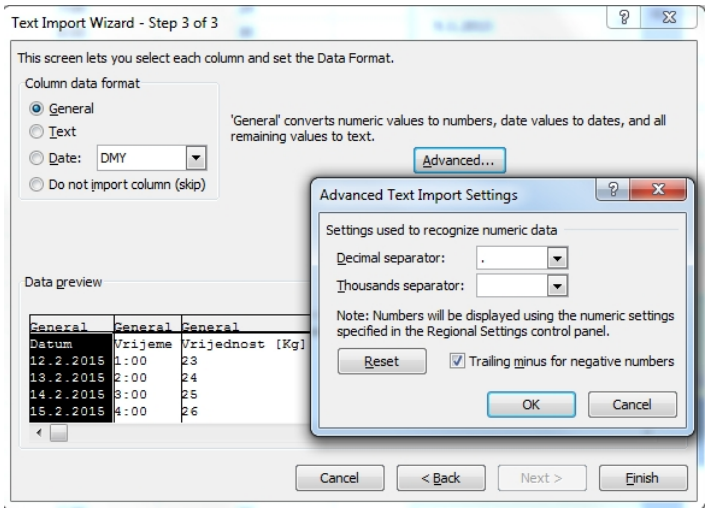
U prvom koraku prikazanom na slici 17 odaberemo datoteku koju želimo uvesti u tablicu te odaberemo *Delimited* kao tip datoteke kako bi mogli interpretirati znak „;“ kao kraj stupca te početak novog. Možemo i odabrati od kojeg retka želimo početi uvoz podataka, u našem slučaju je to 1. Redak. Nakon toga idemo na sljedeći korak.



Slika 18 - Drugi korak

U drugom koraku je potrebno odabrati graničnik koji će nam predstavljati kraj prethodnog i početak novog stupca. U našem je slučaju to „;“ te zbog toga odaberemo „*Semicolon*“.

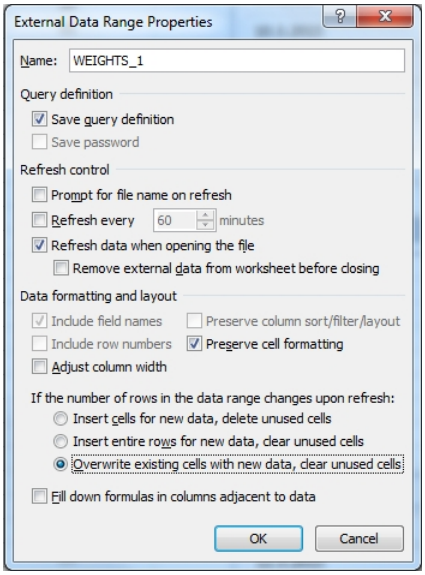
U zadnjem koraku prikazanom na slici 19 odabiremo format podataka te u „*Advanced*“ prozoru definiramo da je simbol za odvajanje decimalnih vrijednosti „.“.



Slika 19 - Treći korak

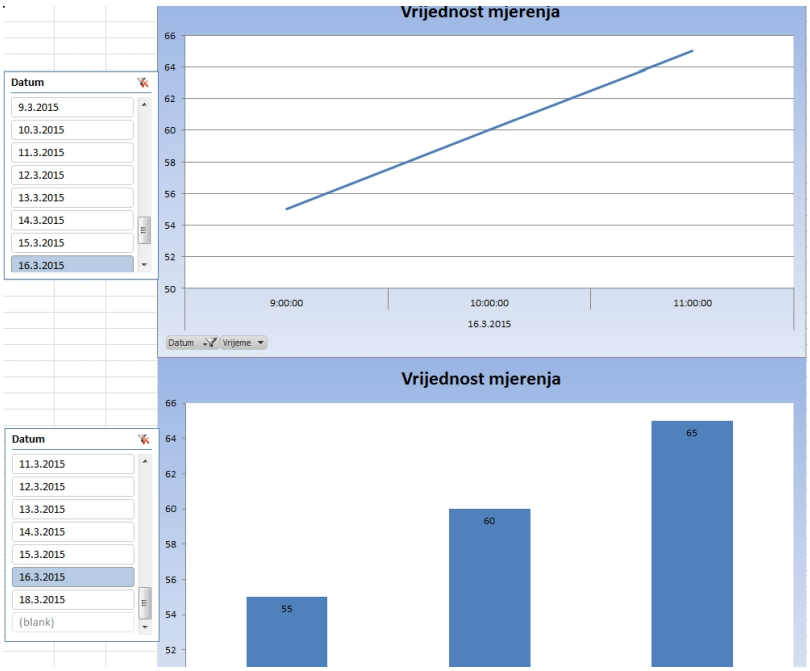
U zadnjem koraku definiramo od koje ćelije želimo da se podaci zapisuju te format ćelija te kako će se ćelije ponašati pri osvježenju podataka te pri pokretanju aplikacije. Želimo da nam se podaci osvježe svakim otvaranjem aplikacije kako ne bi svakim pokretanjem trebali ručno osvježavati. To postižemo odabiranjem „*Refresh data when opening file*“.

Zadnje trebamo odabrati kako želimo da nam se novi podaci ponašaju kada osvježujemo. To je prikazano slikom 20. Ne želimo da nam se samo dodaju novi podaci jer bi to značilo da imamo više istih podataka zapisanih na više mjesta što nije dobro za bazu podataka jer želimo da bude što brža. Zbog toga odabiremo „*Overwrite existing cells with new data, clear unused cells*“. Time prebrisujemo sve stare podatke te stavljamo nove jer nam se .csv file samo nadopunjuje tako da su unutra i podaci koje smo analizirali u neko prethodno vrijeme te novi podaci.



Slika 20 - Definiranje načina osvježavanja podatka

Na slici 21 prikazani su grafovi. U njima je moguće sortirati prikaz po želji kako bi dobili čim čitkiji prikaz podataka. Ako se podaci jako brzo generiraju tada će biti puno podataka u jednom danu pa je korisno da se mogu podaci sortirati i po satu. Datum se može jednostavno birati izbornikom smještenim lijevo od svakog grafa, a raspon datuma ili vremena se može definirati direktno na grafovima izbornicima „datum“ i „vrijeme“.



Slika 21 - Izgled grafova za grafičku analizu podataka

5. Zaključak

Ovaj rad bio je odličan povod za učenje rada na novim platformama. Pošto sam prvi puta radio u svim tim novim razvojnim okruženjima bilo je potrebno puno proučavanja po različitim forumima te *tutorialima*.

Smatram da je ovaj rad prilično dobra osnova da se izradi nekakav komercijalan proizvod temeljen na njemu. Razna poduzeća imaju potrebu za vaganjem te praćenjem dobivenih rezultata. Veliki plus je mogućnost praćenja podataka preko web stranice.

Rad je još daleko od finalne verzije komercijalnog proizvoda ali uz brojna dorađivanja mogao bi to postati.

Relativno lagan zadatak je bio spajanje te čitanje podataka sa vage iako sam imao problem sa stavljanjem vage u vrijednost 0. Problem je bio u oscilacijama rezultata. Oscilacije sam riješio čvršćim spojevima jer kod tako malih struja i napona svaka mala smetnja predstavlja veliku pogrešku u mjerenju.

Da bi se dobio točan rezultat mjerenja potrebno bi bilo još postaviti senzor za mjerenje temperature. Njime bi se u svakom trenutku mogla mjeriti temperatura okoline te prema njoj se vršiti korekcije u prikazu težine jer ovaj senzor koji koristim iako jest precizan, ima mali postotak pogreške s obzirom na temperaturu okoline. Također se pogreška događa kad vaga radi u velikim vremenskim intervalima bez resetiranja.

U Varaždinu, _____

Potpis:

_____

6. Literatura

[1] https://www.arduino.cc/en/Main/Software, dostupno 06.09.2015

[2] https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf, dostupno 06.09.2015

[3] https://en.wikipedia.org/wiki/Piezoresistive_effect, dostupno 06.09.2015

[4] https://www.setweighing.com, dostupno 06.09.2015

[5] https://www.sdcard.org, dostupno 07.09.2015

[6] https://en.wikipedia.org/wiki/Secure_Digital#SDXC, dostupno 08.09.2015

[7] https://github.com/bogde/HX711, dostupno 08.09.2015

[8] https://hr.wikipedia.org/wiki/HTTP, dostupno 08.09.2015

[9] https://hr.wikipedia.org/wiki/Normalizacija_baze_podataka, dostupno 08.09.2015

[10] https://hr.wikipedia.org/wiki/MySQL, dostupno 08.09.2015.

[11] https://hr.wikipedia.org/wiki/API, dostupno 08.09.2015.

Popis slika

Popis tablica

Prilozi

Prilog 1 : Arduino kod za vaganje, pisanje na poslužitelj te SD karticu

```
#include <HX711.h>
#include <Ethernet.h>
#include <SPI.h>
#include <SD.h>
/* ---- vrijeme ---- */
int clockInterrupt = 0; //interrupt 0 je 2. digitalni pin
int pwmOut = 6; //PWM izlaz je na 6. digitlnom pinu

//unesi trenutno vrijeme
int seconds = 0;
int minutes = 21;
int hours = 16;
int days = 8;
int months = 9;
int years = 2015;
int masterClock = 0; //broj rastucih bridova
/* ---- SD kartica ---- */
int SS_SD_CARD = 4;
int SS_ETHERNET = 10;
/* ---- vaga ---- */
// HX711.DOUT      - pin 3
// HX711.PD_SCK    - pin 5
HX711 scale(3, 5);
File myFile;
float current_value = 0;
float previous_value = 0;
/* ---- internet ---- */
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,1,16);
IPAddress hostip (192,168,1,3);
EthernetClient client;

void setup() {
/* ---- vrijeme ---- */
// interrupt, clockCounter() funkcija se poziva na rastuci brid
  attachInterrupt(clockInterrupt, clockCounter, RISING);
  delay(2000);
  Serial.begin(9600);
  delay (1000);
  analogReference(DEFAULT);
  Serial.print("writing to pin: ");
  Serial.println(pwmOut);
  analogWrite(pwmOut, 127);

/* ---- vaga ---- */
  pinMode(3, INPUT_PULLUP);
  scale.set_scale(14605.f);
  delay(1000);
  scale.tare();  // postavljanje vage u 0

/* ----SD kartica---- */
// postavljanje pinova 4 i 10 kao izlaz
  pinMode(SS_SD_CARD, OUTPUT);
  pinMode(SS_ETHERNET, OUTPUT);
  digitalWrite(SS_SD_CARD, HIGH);
  digitalWrite(SS_ETHERNET, HIGH);
  if (!SD.begin(SS_SD_CARD)) {
    Serial.println("Card failed, or not present");
  }
  Serial.println("card initialized.");

/* ----internet----*/
  Ethernet.begin(mac, ip);
  randomSeed(analogRead (0));
  delay(1000);
  Serial.println("connecting...");
}

void loop() {
/* ----zapis za serijsku komunikaciju---- */
  Serial.print("Vrijednost [Kg]:     ");
  serialPrintTime();
  Serial.print(";");
```

```
  Serial.println(scale.get_units(10),1);

/* ----zapis za SD karticu ---- */
  digitalWrite(SS_SD_CARD, LOW);
  delay(10);
  myFile = SD.open("weights.csv", FILE_WRITE);
   if (myFile) {
     SDPrintTime();
     myFile.print(";");
     myFile.print(scale.get_units(10),1);
     myFile.close();
   }
   else {
      Serial.println("error opening weights.csv");
   }
  digitalWrite(SS_SD_CARD, HIGH);

/* ----zapis na server---- */
  digitalWrite(SS_ETHERNET, LOW);
  if (client.connect(hostip, 12233)) {
    Serial.println("connected");
    client.print("GET /weights/insert.php?weight=");
    client.print(scale.get_units(10)*10,1);
    client.print(" HTTP/1.1\r\n");
    client.print("Host: 192.168.1.3\r\n");
    client.print("User-Agent: arduino-ethernet\r\n");
    client.print("Connection: close\r\n\r\n");
    client.println();
 }else {
    Serial.println("connection failed");
 }

  if(client.connected()){
    client.stop();
 }
  digitalWrite(SS_ETHERNET, HIGH);

/* postavljanje vage u sleep mod */
  scale.power_down(); // postavlja HX711 u sleep mod
  delay(5000);
  scale.power_up();
}

// poziva se interruptom 0 na 2. digitalnom pinu kada se pojavi rastuci brid
void clockCounter() {
  // sa svakim rastucim bridom dodaj +1 u masterClock
  masterClock ++;
  if(masterClock == 974) { // frekvencija PWM na pinu 6 je 974 -> 1 sekunda = 974

    seconds ++;
    masterClock = 0;

    if(seconds==60){
     minutes++;
     seconds = 0;

      if(minutes==60){
        hours++;
        minutes = 0;
      }
      if(hours==24){
        days++;
        hours = 0;
      }
    }
  }
return;
}

// funkcija za formiranje oblika zapisa vremena i datuma
void serialPrintTime(){
  if(days<10){
    Serial.print("0");
  }
  Serial.print(days);
  Serial.print(":");
  if (months<10){
    Serial.print("0");
  }
```

```
  Serial.print(months);
  Serial.print(":");
  Serial.print(years);
  Serial.print("  ");
  if(hours<10){
    Serial.print("0");
  }
  Serial.print(hours);
  Serial.print(":");
    if(minutes<10){
      Serial.print("0");
    }
  Serial.print(minutes);
  Serial.print(":");
  if(seconds<10){
    Serial.print("0");
  }
  Serial.print(seconds);
  Serial.print("  ");
}

void SDPrintTime(){
  if(days<10){
    myFile.print("0");
  }
  myFile.print(days);
  myFile.print(":");
  if (months<10){
    myFile.print("0");
  }
  myFile.print(months);
  myFile.print(":");
  myFile.print(years);
  myFile.print(";");
  if(hours<10){
    myFile.print("0");
  }
  myFile.print(hours);
  myFile.print(":");
  if(minutes<10){
    myFile.print("0");
  }
  myFile.print(minutes);
  myFile.print(":");
  if(seconds<10){
    myFile.print("0");
  }
  myFile.print(seconds);
  myFile.print(";");
}
```

Prilog 2: PHP dio koda za spajanje na bazu podataka

```
<?php
    $dbdriver = "mysql";
    $dbname = "weights";
    $dbhost = "localhost";
    $dbuser = "root";
    $dbpass = "";
    $dbport = "3306";

    $dns = $dbdriver . ":host=" . $dbhost . ";dbname=" . $dbname . ";port=" . $dbport;

    $try = 0;
    try{
            $db = new PDO($dns, $dbuser, $dbpass , array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET
NAMES \"UTF8\"", PDO::ATTR_PERSISTENT => false));
    }
    catch(Exception $e){
            echo "fail";

    }
?>
```

Prilog 3: Dohvaćanje podataka iz baze podataka na zahtjev

```
<?php

    include ("ConnectToDatabase.php");

    if (isset($_GET["date"])){
            $queryDate = $_GET["date"];
            $date1 = new DateTime($queryDate . ' 00:00:00.00000');
            $date2 = new DateTime($queryDate . ' 23:59:59.99999');
    }

    $queryString = "SELECT weightValue, dateOfMeasurement FROM weight ";
    if (isset($queryDate)){
            $queryString .= " WHERE dateOfMeasurement BETWEEN (?) AND (?)";
    }
    $queryString .= " ORDER BY dateOfMeasurement DESC";

    $data = array();
    $query = $db->prepare($queryString);
    if (isset($queryDate))
            $query->execute(array($queryDate . ' 00:00:00.00000', $queryDate . '
23:59:59.99999'));
    else
            $query->execute();
   $result = $query->fetchAll();

    foreach ($result as $singleData){
            array_push($data, array("date" => $singleData["dateOfMeasurement"], "weight" =>
$singleData["weightValue"]));
    }

    echo json_encode($data);
?>
```

Prilog 4: Web stranica

```
<?php

    include ("ConnectToDatabase.php");

    $query = $db->prepare("SELECT weightValue, dateOfMeasurement FROM weight ORDER BY
dateOfMeasurement DESC");
    $query->execute();
    $result = $query->fetchAll();

?>

<!DOCTYPE html>
<html>
<head>
    <title>Weight Tracker</title>

</head>
<body>
    <input type="date" id="date" />
    <div id="chart_div">
    </div>
    <table>
            <thead>
                    <tr>
                            <th>Date of Measurement</th>
                            <th>Weight value</th>
                    </tr>
            </thead>
            <tbody>
                    <?php foreach ($result as $singleData) {?>
                            <tr>
                                    <td>
                                            <span><?php echo $singleData["dateOfMeasurement"]
?></span>
                                    </td>
                                    <td>
                                            <span><?php echo
intval($singleData["weightValue"])/10 ?> kilograms</span>
                                    </td>
                            </tr>
```

```
                        <?php } ?>
                </tbody>
        </table>
        <script type="text/javascript" src="jquery-1.11.3.min.js"></script>
        <script type="text/javascript" src="https://www.google.com/jsapi"></script>
        <script type="text/javascript">
                google.load('visualization', '1', {packages: ['corechart', 'line']});
                google.setOnLoadCallback(initializeTable);

                function initializeTable(){
                        var data = new google.visualization.DataTable();
                        data.addColumn('datetime', 'X');
                        data.addColumn('number', 'Weight');
                        var url = "http://localhost:12233/weights/getWeights.php";
                        if ($('#date').val()){
                                url += '?date=' + $('#date').val();
                        }
                        $.get(url, function(weights){
                                weights = JSON.parse(weights);
                                console.log(weights);
                                $.each(weights, function(index, singleWeight){
                                        console.log(singleWeight);
                                        console.log([Date(singleWeight["date"]),
singleWeight["weight"]]);
                                        data.addRows([[new Date(singleWeight["date"]),
parseInt(singleWeight["weight"])/10]]);
                                });
                                console.log(weights);
                                var options = {
                                        hAxis: {
                                                title: 'Date'
                                        },
                                        vAxis: {
                                                title: 'Weight'
                                        },
                                        backgroundColor: '#f1f8e9'
                                }
                                var chart = new
google.visualization.LineChart(document.getElementById('chart_div'));
                                chart.draw(data, options);
                        });
                }
        </script>
        <script type="text/javascript">
                $('#date').change(initializeTable);
        </script>
    </body>
    </html>
```

Prilog 5: Spremanje dobivenih podataka u bazu

```
    <?php

        include ("ConnectToDatabase.php");

        $data = json_decode($_GET["weight"], true);

        $statement = $db->prepare("INSERT INTO weight (weightValue) VALUES ((?))");
        $statement->execute(array($data));

    ?>
```

Prilog 6: Arduino kod kad radi kao poslužitelj i dijeli sadržaj SD kartice

```
    #include <SdFat.h>
    #include <SdFatUtil.h>
    #include <Ethernet.h>
    #include <SPI.h>

    /*----IP konfiguracija */
    byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
    byte ip[] = { 192, 168, 1, 16 };
```

```
EthernetServer server(80);

/*----varijable za SD karticu-----*/
Sd2Card card;
SdVolume volume;
SdFile root;
SdFile file;

// pohrani stringove o greškama u flash memoriju
#define error(s) error_P(PSTR(s))

void error_P(const char* str) {
  PgmPrint("error: ");
  SerialPrintln_P(str);
  if (card.errorCode()) {
    PgmPrint("SD error: ");
    Serial.print(card.errorCode(), HEX);
    Serial.print(',');
    Serial.println(card.errorData(), HEX);
  }
  while(1);
}

void setup() {
  Serial.begin(9600);

  PgmPrint("Free RAM: ");
  Serial.println(FreeRam());

  pinMode(10, OUTPUT);    // postavi SS pin kao izlaz
  digitalWrite(10, HIGH); // ugasi W5100 čip

  if (!card.init(SPI_HALF_SPEED, 4)) error("card.init failed!");

  if (!volume.init(&card)) error("vol.init failed!");

  PgmPrint("Volume is FAT");
  Serial.println(volume.fatType(),DEC);
  Serial.println();

  if (!root.openRoot(&volume)) error("openRoot failed");

  // lista datoteka sa datumima i veličinama
  PgmPrintln("Files found in root:");
  root.ls(LS_DATE | LS_SIZE);
  Serial.println();

  // lista svih direktorija
  PgmPrintln("Files found in all dirs:");
  root.ls(LS_R);

  Serial.println();
  PgmPrintln("Done");
  // pokretanje servera
  Ethernet.begin(mac, ip);
  server.begin();
}

void ListFiles(EthernetClient client, uint8_t flags) {
  dir_t p;
  root.rewind();
  client.println("<ul>");
  while (root.readDir(&p) > 0) {
    if (p.name[0] == DIR_NAME_FREE) break;
    // preskoči izbrisane zapise i zapise za . ili ..
    if (p.name[0] == DIR_NAME_DELETED || p.name[0] == '.') continue;
    // ispisuj samo poddirektorije i datoteke
    if (!DIR_IS_FILE_OR_SUBDIR(&p)) continue;
    client.print("<li><a href=\"");
    for (uint8_t i = 0; i < 11; i++) {
      if (p.name[i] == ' ') continue;
      if (i == 8) {
        client.print('.');
      }
      client.print((char)p.name[i]);
    }
    client.print("\">");
    // ispiši ime datoteke sa mogućim popunjavanjem praznina
    for (uint8_t i = 0; i < 11; i++) {
```

```
      if (p.name[i] == ' ') continue;
      if (i == 8) {
        client.print('.');
      }
      client.print((char)p.name[i]);
    }
    client.print("</a>");
    if (DIR_IS_SUBDIR(&p)) {
      client.print('/');
    }
    // ispiši datum/vrijeme modificiranja datoteke ako je zatraženo
    if (flags & LS_DATE) {
       root.printFatDate(p.lastWriteDate);
       client.print(' ');
       root.printFatTime(p.lastWriteTime);
    }
    // ispiši veličinu ako je zatraženo
    if (!DIR_IS_SUBDIR(&p) && (flags & LS_SIZE)) {
      client.print(' ');
      client.print(p.fileSize);
    }
    client.println("</li>");
  }
  client.println("</ul>");
}

// veličina međuspremnika za svaku liniju
#define BUFSIZ 100

void loop()
{
  char clientline[BUFSIZ];
  int index = 0;

  EthernetClient client = server.available();
  if (client) {
    // http zahtjev završava praznim redom
    boolean current_line_is_blank = true;

    // resetiraj ulazni međuspremnik
    index = 0;

    while (client.connected()) {
      if (client.available()) {
        char c = client.read();

        //ako nije nova linija, dodaj znak u međuspremnik
        if (c != '\n' && c != '\r') {
          clientline[index] = c;
          index++;
          // ako imamo previše znakova u međuspremniku, izbaci jednog
          if (index >= BUFSIZ)
            index = BUFSIZ -1;

          // nastavi čitati podatke
          continue;
        }

        // gledaj da li je poslan \n ili \r što znači da je string gotov
        clientline[index] = 0;

        // ispiši što je klijent poslao
        Serial.println(clientline);

        if (strstr(clientline, "GET / ") != 0) {
          // pošalji standardno zaglavlje HTTTP odgovora
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println();

          // ispiši sve fileove
          client.println("<h2>Files:</h2>");
          ListFiles(client, LS_SIZE);
        } else if (strstr(clientline, "GET /") != 0) {

          char *filename;

          filename = clientline + 5;
          (strstr(clientline, " HTTP"))[0] = 0;
```

```
            // ispiši datoteku koja je odabrana
            Serial.println(filename);

            if (! file.open(&root, filename, O_READ)) {
              client.println("HTTP/1.1 404 Not Found");
              client.println("Content-Type: text/html");
              client.println();
              client.println("<h2>File Not Found!</h2>");
              break;
            }

            Serial.println("Opened!");

            client.println("HTTP/1.1 200 OK");
            client.println("Content-Type: text/plain");
            client.println();

            int16_t c;
            while ((c = file.read()) > 0) {
                client.print((char)c);
            }
            file.close();
          } else {
            // Sve ostalo je error 404 - file not found
            client.println("HTTP/1.1 404 Not Found");
            client.println("Content-Type: text/html");
            client.println();
            client.println("<h2>File Not Found!</h2>");
          }
          break;
        }
      }
      delay(1);
      client.stop();
    }
  }
```

Prilog 7: PHP dio koda za spajanje na arduino

```
<?php

    $curl = curl_init();

    curl_setopt_array($curl, array(
            CURLOPT_RETURNTRANSFER => 1,
            CURLOPT_URL => 'http://192.168.1.16'
    ));

    $resp = curl_exec($curl);

    curl_close($curl);

    echo str_replace("href=\"","class=\"arduinoFileLink\"
href=\"getFileFromArduino.php?filename=",$resp);
?>
```

Prilog 8: PHP dio koda za biranje Arduinove datoteke

```
<?php
    $filename = $_GET["filename"];
    $curl = curl_init();

    curl_setopt_array($curl, array(
    CURLOPT_RETURNTRANSFER => 1,
    CURLOPT_URL => 'http://192.168.1.16/' . $filename
    ));

    $resp = curl_exec($curl);

    curl_close($curl);

    echo $resp;
?>
```

Prilog 9: Web stranica za dobivanje podataka iz SD kartice i njihovu vizualizaciju

```
<!DOCTYPE html>
<html>
<head>
    <title>Weight Tracker</title>

</head>
<body>
    <button type="button" id="getFromArduino">Check what Arduino has to offer!</button>
    <div id="arduinoResponse"></div>
    <div id="chart_div">
    </div>
    <table id="weightTable">
            <thead>
                    <tr>
                            <th>Date of Measurement</th>
                            <th>Weight value</th>
                    </tr>
            </thead>
            <tbody>
            </tbody>
    </table>
    <script type="text/javascript" src="jquery-1.11.3.min.js"></script>
    <script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
            google.load('visualization', '1', {packages: ['corechart', 'line']});

            function initializeTable(url){
                    var data = new google.visualization.DataTable();
                    data.addColumn('datetime', 'X');
                    data.addColumn('number', 'Weight');

                    $.get(url, function(weights){
                            weights = weights.replace(/ /g, '');
                            weights = weights.split('\n');
                            $('#weightTable > tbody').html('');
                            $.each(weights, function(index, singleWeight){
                                    if (!singleWeight) return true;
                                    var row = singleWeight.split(';');
                                    var dateCol = row[0].split(':');
                                    var timeCol = row[1].split(':');
                                    var weight = parseFloat(row[2].replace(/,/g, '.'));

                                    var date = new Date(dateCol[2], dateCol[1], dateCol[0],
timeCol[0], timeCol[1], timeCol[2]);

                                    data.addRows([[date, weight]]);
                                    $('#weightTable > tbody:last-child').append('<tr><td>' +
date.getDay() + '.' + (date.getMonth()+1).toString() + '.' + date.getFullYear().toString() +
'.</td><td>' + weight + '</td></tr>');
                            });
                            var options = {
                                    hAxis: {
                                            title: 'Date'
                                    },
                                    vAxis: {
                                            title: 'Weight'
                                    },
                                    backgroundColor: '#f1f8e9'
                            }
                            var chart = new
google.visualization.LineChart(document.getElementById('chart_div'));
                            chart.draw(data, options);
                    });
            }
    </script>
    <script type="text/javascript">
            $("#getFromArduino").click(function(){
                    $.get('checkArduino.php', function(response){
                            $('#arduinoResponse').html(response);

                            $('.arduinoFileLink').click(function(event){
                                    var element = $(event.target);
                                    var link = element.attr('href');
                                    initializeTable(link);
```

```
                                return false;
                            });
                        })
                });
        </script>
</body>
</html>
```