

Generator naključnih števil z uporabo kaotičnih sistemov

Jakob Lavrič, Matjaž Pogačnik
Vhodno izhodne naprave

Table of Contents

Kaotični sistemi.....	3
Dvojno nihalo	3
Chua's circuit	3
Uporaba kaotičnih sistemov za generator naključnih števil.....	5
Generiranje naključnih števil.....	5
Zajemanje vrednosti	5
Tekoči povprečevalnik.....	6
Sestavljanje števil.....	6
Algoritem za kalibracijo	7
Obdelava generiranih števil.....	8
Analiza generiranih števil	8
Entropija generiranih števil	9
Hitrost generiranja	10
Nadgradnja vezja	10
Zaključek	13

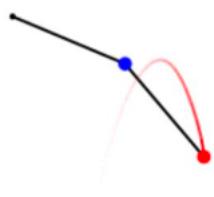
Kaotični sistemi

Kaotični sistem je definiran kot deterministični sistem, ki je težko predvidljiv. Deterministični pomeni, da njegovo trenutno stanje popolnoma definira njegova stanja kadarkoli v prihodnosti, na primer, sistem diferencialnih enačb za dvojno nihalo. Torej lahko popolnoma predvidimo prihodnja stanja sistema, če vemo njegovo trenutno stanje. Kaotičnost oziroma nepredvidljivost pa izvira iz natančnosti napovedi prihodnjega stanja, saj je lahko obnašanje sistema v prihodnosti močno občutljivo na njegovo trenutno stanje. Ker je praktično nemogoče stanje sistema opisati s popolno natančnostjo, je tako prihodnost sistema še toliko težja oziroma nemogoča za napovedati. Take sisteme lahko uporabimo za generator naključnih števil, kjer je nepredvidljivost naslednjega generiranega števila lahko zelo pomembna.

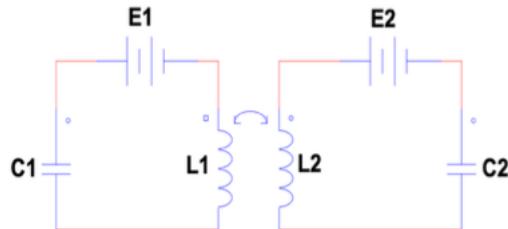
Dvojno nihalo

Klasičen primer kaotičnega sistema je dvojno nihalo, ki je bil tudi motivacija tega projekta. Dvojno nihalo dobimo, če na konec osnovnega nitnega nihala, s periodičnim nihanjem, prirdimo še eno nihalo. Gibanje takega sistema ni več periodično, temveč kaotično.

Ker je dvojno nihalo fizično težko realizirati brez dodatnega trenja, lahko dvojno nihalo realiziramo tudi z električnim vezjem, kjer s transformatorjem povežemo dva električna nihajna kroga. Izkaže pa se, da je realizacija takega vezja prav tako precej zahtevna zaradi idealnega transformatorja, ki je gradnik električnega dvojnega nihala.



Slika 1: Dvojno nitno nihalo

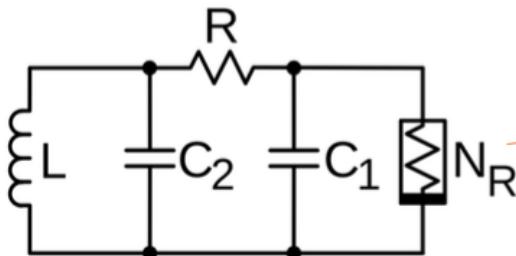


Slika 2: Dvojno električno nihalo

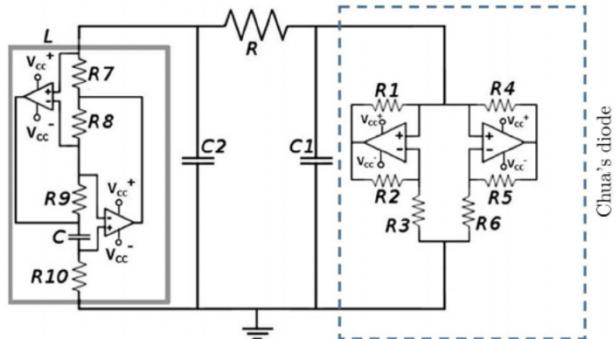
Chua's circuit

Idealen transformator lahko realiziramo s pomočjo komponent, kot so operacijski ojačevalnik, VCVS (voltage controlled voltage source), VCCS (voltage controlled current source) ... Obstaja pa minimalno vezje, ki je glede na pogoje, ki morajo veljati za kaotičen sistem, minimalno. Za vezje sestavljeno iz osnovnih elektronskih komponent kot so uporniki, kondenzatorji in induktorji, so te pogoji uporaba enega ali večih nelinearnih komponent, enega ali večih lokalno aktivnih uporov in treh ali večih komponent, ki hranijo energijo. Lokalno aktivni upor predstavlja upor, ki je lahko negativen in služi kot vir energije.

Nelinearno komponento in lokalno aktivni upor v Chua's circuit predstavlja Chua's diode (NR), ki jo lahko realiziramo na primer z uporabo operacijskih ojačevalnikov.



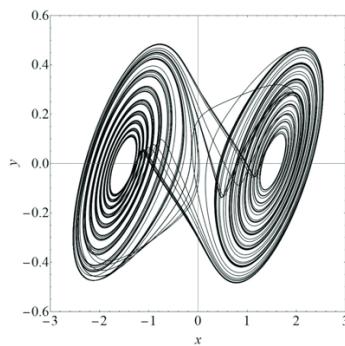
Slika 4: Shema osnovnega vezja



Slika 3: Shema vezja iz uresničljivih komponent

Za induktor L bomo prav tako uporabili realizacijo s pomočjo operacijskih ojačevalnikov in nastavljenih uporov. Tako bomo dobili nastavljen induktor, ki nam bo omogočal mnogo lažje nadzorovanje kaotičnega sistema. Kaotični sistemi imajo namreč lahko stanja v katerih preidejo v periodično delovanje. Elektronsko vezje je v periodičnem stanju večino časa, kaotično pa postane samo v določenih specifičnih parametrih komponent.

Kaotičnost sistema lahko spremljamo s pomočjo osciloskopa, kjer spremljamo napetosti na kondenzatorji C1 in C2. Če te napetosti izrisujemo na x in y os, lahko vidimo gibanje stanj sistema skozi čas. V tem načinu namesto šuma vidimo definirane krivulje, okrog katerih se gibajo stanja. Tem oblikam krivulj pravimo atraktorji in opisujejo stanja, proti katerim konvergirajo različna poljubna stanja sistema skozi čas. Ker jih lahko tako opišemo, bodo majhne razlike med začetnimi stanji res povzročile velike razlike med kasnejšimi stanji, vendar se bosta te stanji nekoč spet srečali. Taki sistemi so lahko torej kljub lokalni nestabilnosti pri majhnih razlikah v izhodiščnih pogojih, vseeno globalno stabilni in se jih da opisati. Če bi dvojno nihalo dušili oziroma upoštevali trenje, bi njegovo gibanje prav tako lahko opisali s krivuljo, ki bi bila v tem primeru ena sama točka, ki bi v končni fazi predstavljal njegovo mirujoče stanje, v osnovi, pa dvojno nihalo take podmnožice stanj nima. V našem vezju bomo za periodično stanje opazili elipse, pri določenih pogojih pa lahko vidimo atraktor značilen za ta kaotičen sistem – double scroll.



Slika 5: Double scroll



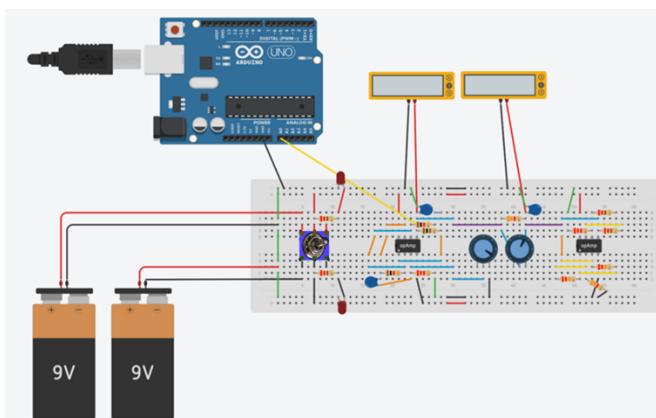
Slika 7: Periodično obnašanje vezja



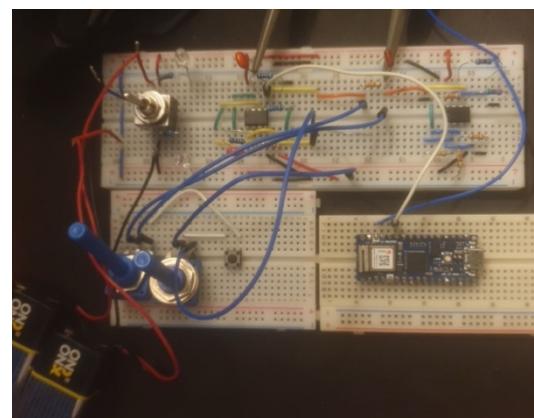
Slika 6: Kaotično obnašanje sistema (Double scroll)

Uporaba kaotičnih sistemov za generator naključnih števil

Zaradi nepredvidljivosti, oziroma kaotičnega obnašanja stanj sistema, lahko le-te zajemamo in jih izkoristimo za generator naključnih števil. Ob vsakem zagonu sistema, se bo ta inicializiral na naključno stanje zaradi vedno prisotnega šuma, ki se zaradi karakteristike kaotičnega sistema odraža s vedno drugim, nepredvidljivim začetnim stanjem.



Slika 9: Zajemanje stanj Chua's circuit z Arduinom



Slika 8: Realizacija vezja

Generiranje naključnih števil

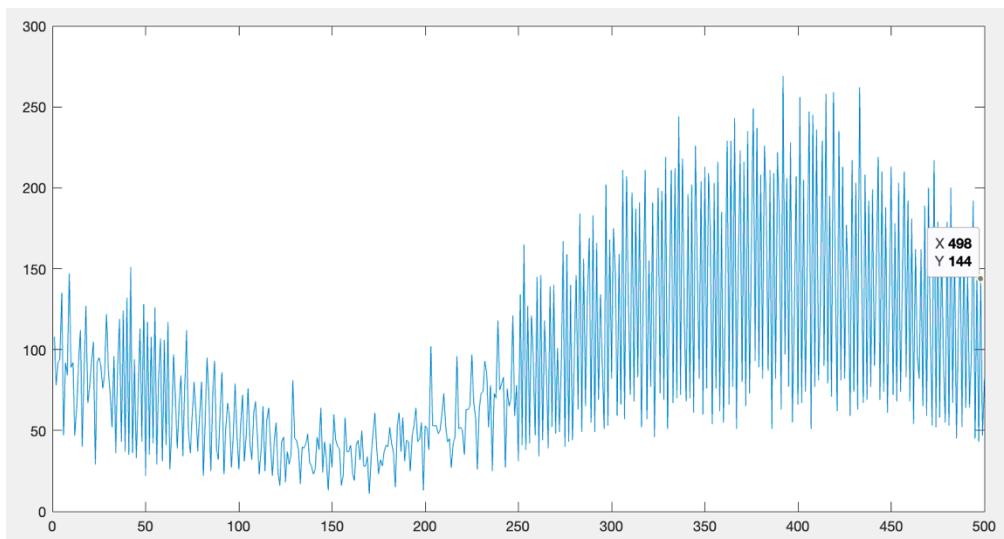
Zajemanje vrednosti

Ko imamo vezje nastavljeno na parametre potrebne za kaotično delovanje, je dovolj, da zajemamo napetost samo na enem kondenzatorju. V tem projektu je bil uporabljen Arduino Nano 33 IoT, vrednosti pa so bile zajete s pomočjo ADC pretvornika. Tako so bile z vezja zajete 10bitne vrednosti.

Zaradi nivojev na katerih operira vezje, so bile zajete vrednosti samo na intervalu približno 0-150 (od 1024), kjer so bile za generiranje števil vzete samo pozitivne vrednosti, saj ADC pretvornik na Arduinu ne zajema negativnih napetosti. Za preslikavo teh vrednosti na poljuben interval je to maksimalno 150 različnih števil, kar za naš generator ni dovolj.

Tekoči povprečevalnik

Da iz našega generatorja dobimo več vrednosti, lahko vrednosti hranimo v bufferju, kjer kot rezultat vračamo vsoto elementov bufferja, po nekem modulu. Pri tem se pojavit dva problema. Modulo, po katerem bomo vračali rezultat mora možne vrednosti vsot od 0 do 1300 in vzamemo modulo 1000, bodo vrednosti od 0-300 pri enako verjetnih vhodnih vrednostih dvakrat bolj verjetne kot vrednosti 301-1000. Drug problem pa je ta, da se vsota obnaša nekoliko kot povprečenje izhoda iz vezja, kar se vidi iz porazdelitve generiranih števil, ki sledijo neki krivulji. Domnevno je to zaradi oblike (atraktorja), ki ji sledijo napetosti v vezju, kljub njihovi naključnosti. S povprečenjem bomo tako dobili izgubili pravo naključnost zajetih vrednosti.



Slika 10: Frekvence števil 0-500 pri tekočem povprečevalniku

Sestavljanje števil

Drug pristop je, da števila gradimo iz večih zajetih vrednosti. Če želimo generirati 32bitno celo število, lahko to število sestavimo iz poljubne permutacije $2^*10\text{bitov} + 2\text{bita}$. Za to bi potrebovali 3 10bitne zajeme. Ker so zajete vrednosti dejansko samo na intervalu od 0-150, imamo naključnih samo nekaj več kot prvih 7bitov. Zato lahko število sestavimo iz 7bitnih blokov. Pri n -bitnih blokih ne smemo uporabiti vrednosti večjih od $2^n - 1$, saj bi prišlo do prej omenjene težave z verjetnostmi pri računanju po modulu.

Hitrost generiranja tako sestavljenih števil je večja, če izkoriščamo celoten obseg ADC pretvornika, kar lahko dosežemo s tem, da peljemo signal skozi ne-invertirajoči operacijski ojačevalnik. Če želimo trenutno napetost 1,8V ojačati na 5V, moramo to skalirati za faktor 2,78. Pravilno izbrane vrednosti upora na vhodu in feedback upora

določajo ta faktor ojačanja. Vhodni signal povežemo na ne-invertirajoči vhod operacijskega ojačevalnika, izhod pa na analogni vhod Arduina.

Za bolj splošen interval, ki ga dobimo iz ADC pretvornika, lahko n-bitno število a gradimo iz n zajetih vrednosti, tako da je i -ti bit generiranega števila a_i enak 1, če je zajeta vrednost v_i večja od neke meje t , oziroma 0 v nasprotnem primeru.

$$a_i = \begin{cases} 1 & , v_i > t \\ 0 & , \text{sicer} \end{cases} \quad a = a_0 a_1 a_2 \cdots a_{n-1}$$

Za ustrezno porazdelitev generiranih števil moramo t določiti tako, da bo v generiranih številih zastopanost 1 in 0 približno enaka. Za to lahko uporabimo bisekcijo:

Algoritem za kalibracijo

V spodnjem algoritmu vsakih 10 sekund preverimo razmerje 1 in 0. Kalibracijo končamo, ko je po treh iteracijah zastopanost 0 in 1 še vedno enaka. Če je zastopanost 1 večja od 0, bomo bisektivno interval $[a, b]$ premaknili na $[(a + b)/2, b]$ oziroma na $[a, (a + b)/2]$ v nasprotnem primeru.

Za mejo uporabimo sredino intervala $t = (a + b)/2$.

```
function calibrate(a, b):
    check = 3

    while check != 0:
        voltage = analogRead(analogPin)

        if voltage != 0:
            if voltage > (a+b) / 2:
                ones++
            else:
                zeros++
            count++

        if 10 seconds passed:
            if round(zeros*100)/count) == round((ones*100)/count):
                check--
            else:
                check = 3

            if o > z:
                a = (a+b) / 2
            else:
                b = (a+b) / 2

        ones = 0
        zeros = 0
        count = 0

    return t = (a+b) / 2
```

Obdelava generiranih števil

Za nadaljno programsko obdelavo števil iz Arduina po serialu pošiljamo zaporedje generiranih števil a iz intervala $[0,1]$ z dvojno natančnostjo. Števila s tega intervala dobimo tako, da generirano 32bitno število delimo z 2^{32} . Zaporedje potem lahko zajemamo s Pythonom, kjer lahko lažje sprogramiramo uporabniški vmesnik oziroma analiziramo generirana števila. Ena glavnih funkcionalnosti za uporabnika je določanje intervala generiranja števil. Števila iz intervala $[s, e]$ tako generiramo po formuli

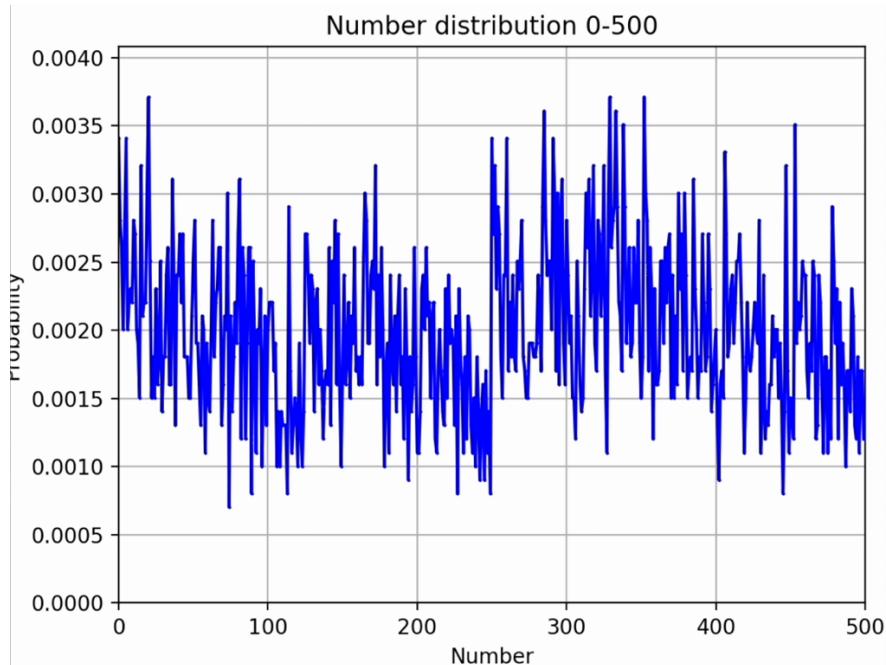
$$d = a(e - s) + s$$

Če sestavljamo 32bitna števila je naš generator sposoben generirati 2^{32} števil. Če bomo generirana števila preslikovali na večji interval, moramo ustrezeno sestavljati števila z več biti.

Analiza generiranih števil

Kriterijev za dober generator naključnih števil je več. Za približno enako zastopanost 0 in 1 smo poskrbeli, poleg tega pa smo števila generirali iz zajetih napetosti iz vezja, ki so po definiciji nepredvidljive. Želimo da so generirana števila približno enakomerno porazdeljena, vendar vseeno nepredvidljivo generirana. Skozi čas tako pričakujemo, da bodo vse relativne frekvence števil v čedalje manjšem intervalu okrog vrednosti $p = 1/n$. Naključno smo generirali števila od 0-500 in njihove relativne frekvence prikazali na grafu.

$$p = \frac{1}{500} = 0,002$$



Slika 11: Relativne frekvence po 10000 generiranih števil

Entropija generiranih števil

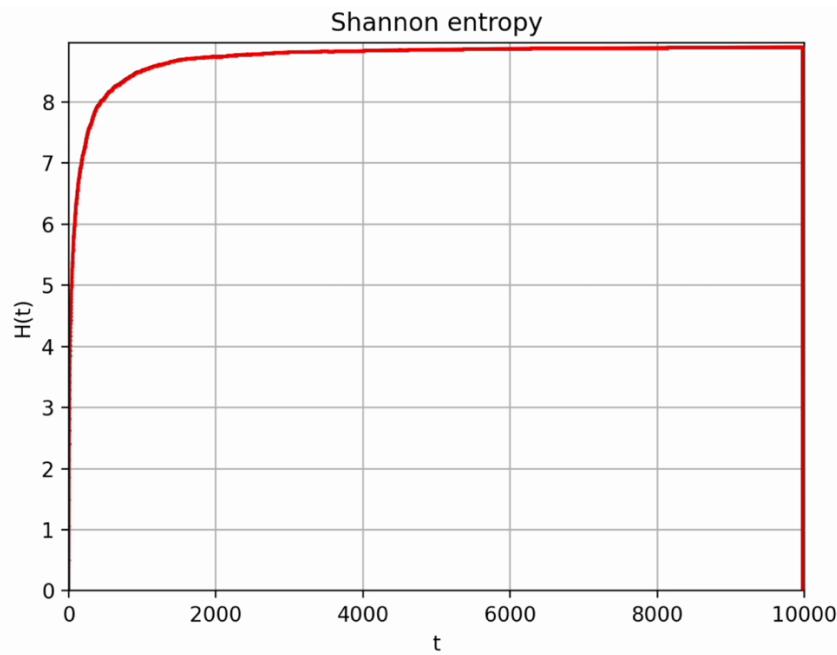
Pogoj za uspešen generator naključnih števil je visoka mera entropije. V meritvah smo uporabili shannonovo entropijo, ki je definirana kot

$$H_r = \sum_{i=0}^n -p_i \log_r p_i$$

Pove nam povprečno količino informacije, ki jo pridobimo z vsakim novim generiranim številom. Shannonova entropija je največja, ko so vse relativne frekvence, oziroma verjetnosti p_i enake. Generator naključnih števil mora imeti torej čim večjo entropijo. Parameter r v enačbi nam predstavlja bazo v kateri bomo izrazili količino informacije. Če vzamemo $r = 2$ bomo informacijo izrazili v bitih. Maksimalna entropija za naš primer je torej

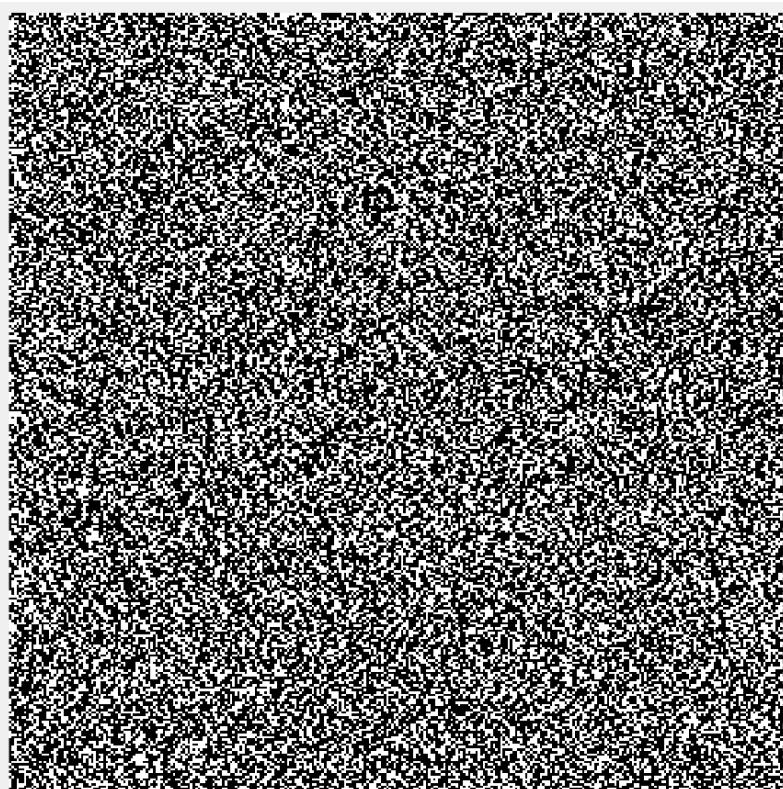
$$H = -\log_2 p = -\log_2 0,002 = 8,9658$$

Na spodnjem grafu entropije tekom generiranja 10000 števil vidimo, da se zelo hitro močno približamo maksimalni entropiji. Maksimalne entropije pa zaradi naključnega generiranja v splošnem nebomo nikoli dosegli.



Slika 12: Shannonova entropija pri generiranju 10000 številih

Še en nekoliko manj zanesljiv test je nizanje generiranih vrednosti v matriko, kjer vizualno preverjamo, če je opazen kakšen vzorec. Na spodnji sliki vzorca ni videti.



Slika 13: Matrika zajetih 0 in 1

Hitrost generiranja

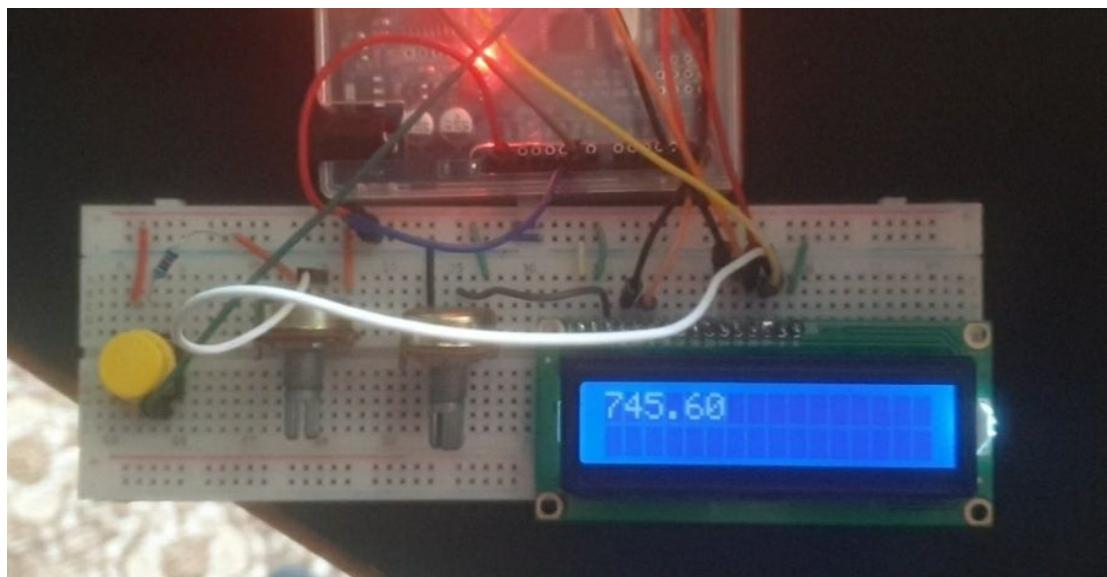
Hitrost generiranja ni konstantna zaradi filtriranja negativnih napetosti, ki jih ADC pretvornik pretvori v vrednost 0. Pri grajenju števil iz 32 zajemov je generator v eni minuti generiral 919 števil, kar znaša 15.32 števil na sekundo.

Nadgradnja vezja

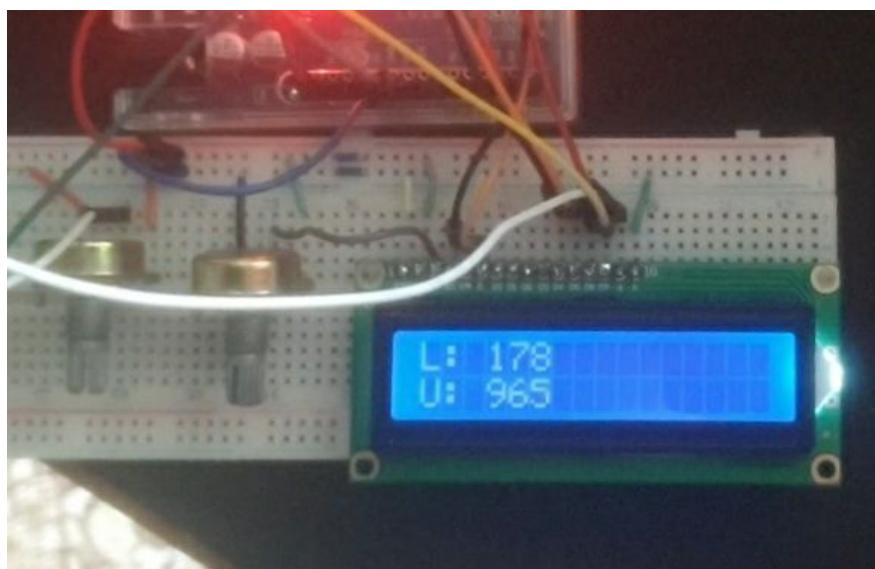
Vezje lahko za voljo kompaktnosti naredimo avtonomno, tj. da generator deluje kot samostojna naprava, brez uporabe računalnika. Za osnovno delovanje vezja je edina informacija, ki jo mora vezje pridobiti od uporabnika, interval na katerem bo generator generiral števila. Za nadzor delovanja lahko uporabimo gumb, s katerim uporabnik zahteva novo število in gumb, ki sproži kalibracijo.

Za nastavljanje intervala bomo uporabili rotacijski enkoder z gumbom. Uporabnik z gumbom prestavlja med izbiro začetka intervala in konca intervala. Z vrtenjem enkoderja nato nastavi vrednost.

Nastavljanje intervala, potek kalibracije in generirana števila lahko uporabnik vidi na LCD displayu.



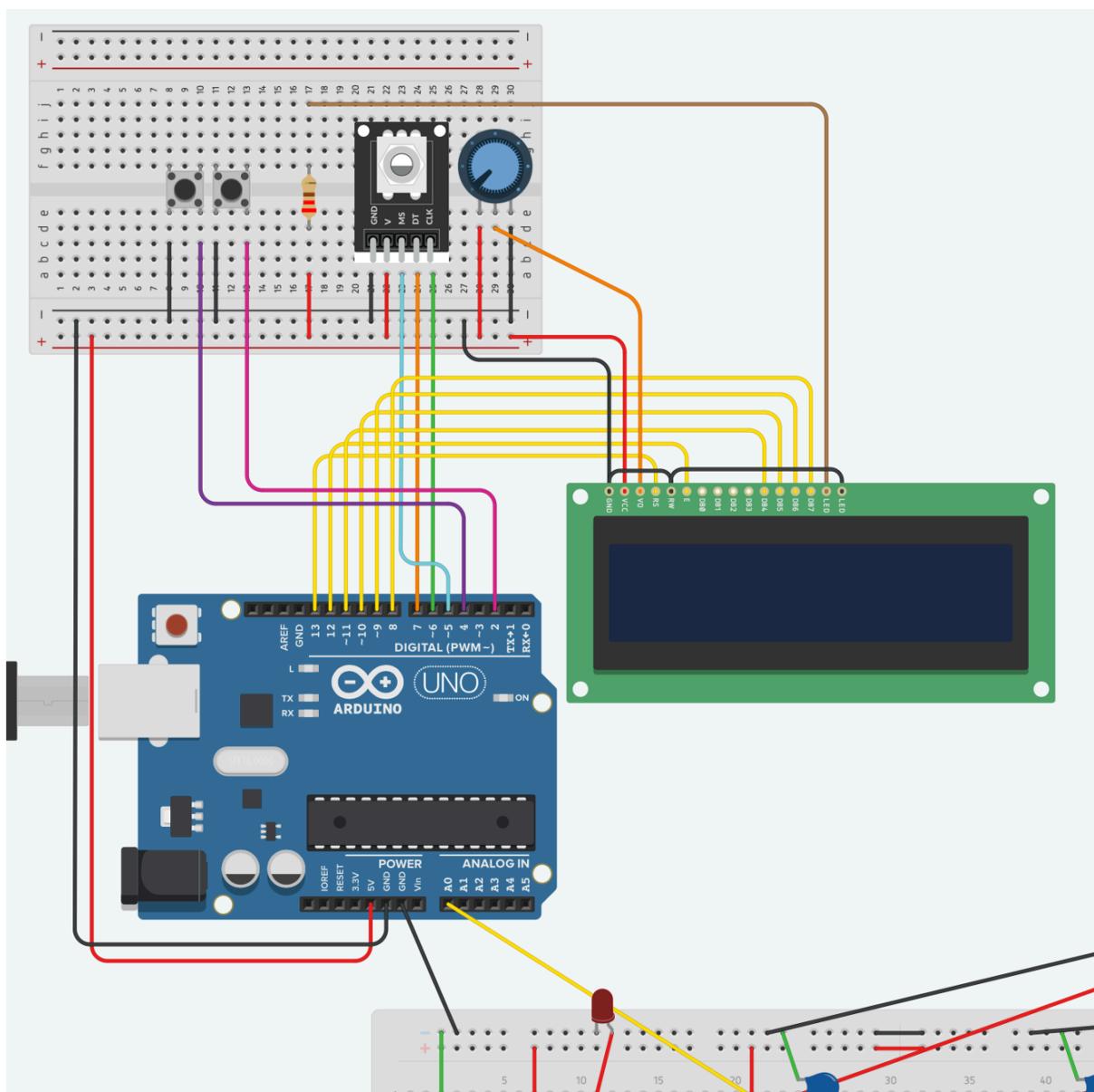
Slika 14: Generiranje števil brez računalnika



Slika 15: Nastavljanje intervala



Slika 16: Kalibracija brez računalnika



Slika 17: Shema nadrgajenega dela vezja

Zaključek

Uspešno smo ustvarili vezje, ki se obnaša kaotično, kar smo preverili z opazovanjem značilnega atraktorja. Stanje vezja smo nato zajemali z mikrokrumilnikom Arduino, kjer smo podatke obdelali in jih pošiljali na serijski izhod za nadaljno obdelavo v Pythonu, kasneje pa smo generator nadgradili za uporabo brez računalnika.

Generator s pomočjo Chua's circuit je sposoben naključno, vendar enakomerno generirati števila iz želenega intervala, kar je bil cilj projekta. Nekaj osnovnih meril za dober generator naključnih števil smo preverili tudi s pomočjo Shannonove entropije in opazovnjem porazdelitve števil.

Za generiranje naključnih števil želimo vezje v kaotičnem stanju, kar je lahko malce nadležno pri vezju narejenem na breadboardu. Zaradi manj solidnih povezav lahko vezje prehaja iz kaotičnega stanja nazaj v predvidljivo oscilacijo že ob najmanjših premikih. Ker vezje kontroliramo z dvema potenciometroma je važno, da sta vsaj ta dva čim bolj solidno pritrjena in kvalitetna, zato bi bilo bolje vezje natisniti ali pa ga narediti na protoboardu, kjer so povezave bolj stabilne.

Posnetki generatorja in koda so objavljeni na <https://github.com/MAZI2/Chuas-circuit-Random-number-generator>

Literatura

- [1] Gergely Vadai, Zoltán Gingl, & János Mellár. (2012). Real-time demonstration of the main characteristics of chaos in the motion of a real double pendulum. European Journal of Physics, 33, 907. DOI: 10.1088/0143-0807/33/4/907
- [2] Chen, J. (2008, February 5). Chaos from Simplicity: An Introduction to the Double Pendulum. In Chapter 8
- [3] Lorenz system | Wikipedia. Dostopno na: https://en.wikipedia.org/wiki/Lorenz_system#Analysis
- [4] Attractor | Wikipedia, Dostopno na: https://en.wikipedia.org/wiki/Attractor#Strange_attractor
- [5] Double pendulum | Wikipedia, Dostopno na: https://en.wikipedia.org/wiki/Double_pendulum
- [6] Butterfly effect | Wikipedia, Dostopno na: https://en.wikipedia.org/wiki/Butterfly_effect
- [7] Chaos theory | Wikipedia, Dostopno na: https://en.wikipedia.org/wiki/Chaos_theory
- [8] Hasan, M., Saha, C., Rahman, M. M., Sarker, M. R. I., & Aditya, S. K. Balancing of an Inverted Pendulum Using PD Controller. (June 2012). Dhaka University Journal of Science, 1(1), 1-6. DOI: 10.3329/dujs.v60i1.10348
- [9] AMS OSRAM Group. AS5030: 8-Bit Programmable High-Speed Magnetic Rotary Encoder Documentation, Dostopno na: <https://look.ams-osram.com/m/2cbef81b5d2c2b54/original/AS5030-DS000199.pdf>