Pulse induction metal detector

using Raspberry Pi Pico RP2040 microcontroler

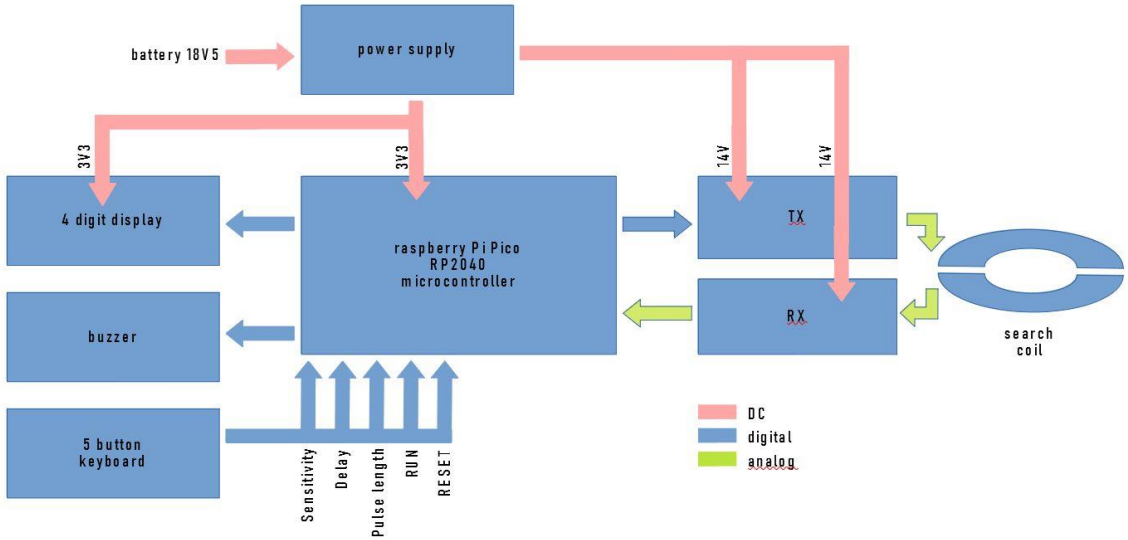
v2_03042021

1. Principle of pulse induction metal detection

Pulse induction metal detection relies on the fact that when metal is in a changing magnetic field, eddy currents are induced in the metal. These eddy currents form complete circuits which are equivalent to electro-magnets which generate their own magnetic fields. As these induced magnetic fields fluctuate, they also induce current in the originating inductor (searching coil). If the searching coil in this circuit is driven with a current pulse, when the pulse ends, current will continue to flow in the searcing coil and in this case the current will decay because there is a resistive load on the antenna. As the current decays, the magnetic field will also collapse. The effect of external metal in the magnetic field of the searching coil and the resulting fields produced by induced eddy currents is to prolong the decay of the current in the searching coil. We can detect metal near the searcing coil by noticing when the decay period starts to extend and measuring voltage difference against normal decay.

Principal advantage of this kind of metal detector is hardware simplicity and simple tuning operations. The sensitivity is also good. As drawback, is difficult to implement a method to discriminate metal type.

2. Block diagram



3. Schematic and operation

For schematic and component list , download the file **xaj_v2_03042021 schematic.pdf**

The analog part of the schematic is classic, is inspired by SPIRIT PI 2.0 ESP32 (N.E.C.O) schematic and often used by a lot of people who are manufacturing their own metal detectors. But idea was to replace the digital part (usualy ESP32, ATMega8 or Arduino Uno boards) with Raspberry Pi Pico based on RP2040 microcontroller.

What is Raspberry Pi Pico ? I am sure that many people already know the answer and they use the board for different projects. Here are only some important features of the board:

Form factor	21 mm × 51 mm
CPU	Dual-core Arm Cortex-M0+ @ 133MHz
Memory	264KB on-chip SRAM; 2MB on-board QSPI Flash
Interfacing	26 GPIO pins, including 3 analogue inputs
Peripherals	2 × UART, 2 × SPI controllers, 2 × I2C controllers, 16 × PWM channels, 1 × USB controller and PHY, with host and device support, 8 × PIO state machine
Input power	1.8–5.5V DC
Operating temperature	-20°C to +85°C

For better understanding of this hardware, integration into your projects and programming , please visit official site:
https://www.raspberrypi.org/products/raspberry-pi-pico/
https://www.raspberrypi.org/documentation/rp2040/getting-started/

So, presuming that you are documented about Raspberry Pi Pico, let's continue our project and let's make a short description of operating mode. PICO will generate at GPIO16 pin a 3,3v pulse with a certain length. This pulse will control a driver using Q1 and Q2 and will be applied to Q3, witch has the searching coil in series with drain. The pulse will produce a magnetic field generated by the searching coil, and becose of eddy currents, the target will produce a very low corresponding voltage at the coil terminals.

The mode this signal decay depends of target characteristics (form, type, material) and the distance. This is the signal we need to analyze, so it is supressed by diodes D1 and D2 and passed to an operational amplifier U1. After amplification, the analog signal is read by analog input (channel 1) of Raspberry Pi Pico – GPIO27 and analized by the software. This poccess is repeatedly for many times per second.

4. The software

The software use Micropyhon for Raspberry Pi Pico . Below is the code and a flowchart of the application. For better resolution, download and check file **xaj_v2_03042021 fwchart.pdf** . This will help understand how the application works and also can help for future improvements. This version can only detect the presence of metal targets, but no discrimination is done.

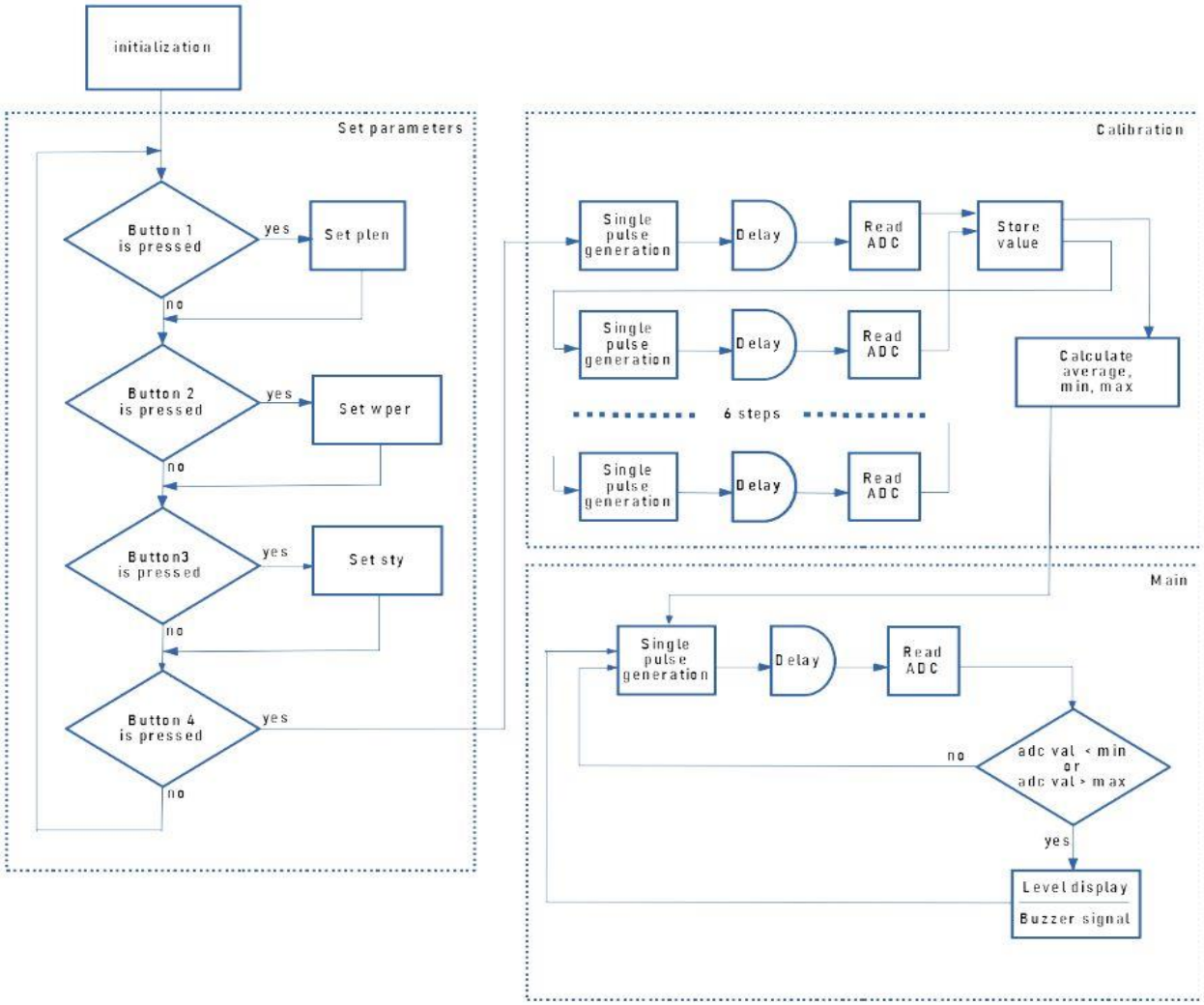
Here is the code:

```
"""
Copyright (c) 2021 Daniel Mihai

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
"""

from machine import Pin, PWM
from machine import ADC
import utime
import tm1637

global avss, maxss, minss
global lstplen, lstwper, lststy, lstfrq
global plen, wper, sty, frq
```

```
global focusplen, focuswper, focussty, focusfrq

focusplen=7
focuswper=4
focussty=3
focusfrq=3

lstplen=[50, 75, 100, 150, 200, 250, 300, 350, 400, 500]
lstwper=[20, 30, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300]
lststy=[0, 50, 100, 150, 200, 250, 300, 350, 400, 500, 700]
lstfrq=[20, 30, 40, 50, 60, 70, 80, 90, 100, 120, 140, 160]

plen=lstplen[focusplen]
wper=lstwper[focuswper]
sty=lststy[focussty]
frq=lstfrq[focusfrq]

rd=machine.ADC(1)
tm = tm1637.TM1637(clk=Pin(5), dio=Pin(4))
tm.brightness(0)

tm.show("----")
impuls=Pin(16, Pin.OUT)
buzzer=PWM(Pin(15))

btplen = Pin(10, Pin.IN, Pin.PULL_UP)
btwper = Pin(11, Pin.IN, Pin.PULL_UP)
btsty = Pin(12, Pin.IN, Pin.PULL_UP)
btop = Pin(13, Pin.IN, Pin.PULL_UP)
bfrq= Pin(9, Pin.IN, Pin.PULL_UP)

while True:
    if not bfrq.value():
        if focusfrq == 11:
            focusfrq=-1
        focusfrq=focusfrq+1
        frq=lstfrq[focusfrq]
        tm.number(frq)
        utime.sleep_ms(500)
    if not btplen.value():
        if focusplen == 9:
            focusplen=-1
        focusplen=focusplen+1
        plen=lstplen[focusplen]
        tm.number(plen)
        utime.sleep_ms(500)
    if not btwper.value():
        if focuswper == 12:
            focuswper=-1
        focuswper=focuswper+1
        wper=lstwper[focuswper]
        tm.number(wper)
        utime.sleep_ms(500)
    if not btsty.value():
        if focussty == 10:
            focussty=-1
        focussty=focussty+1
```

```
    sty=lststy[focussty]
    tm.number(sty)
    utime.sleep_ms(500)
  if not btop.value():
    plen=lstplen[focusplen]
    wper=lstwper[focuswper]
    sty=lststy[focussty]
    frq=lstfrq[focusfrq]
    tm.show("SCAL")
    buzzer.freq(1000)
    buzzer.duty_u16(1000)
    utime.sleep_us(500000)
    buzzer.duty_u16(0)
    lstcal=[]
    for i in range (0,6):
      impuls.value(1)
      utime.sleep_us(plen)
      impuls.value(0)
      utime.sleep_us(wper)
      vadcal=rd.read_u16()
      lstcal.append(vadcal)
      utime.sleep_us(frq*1000)
    avss=round(sum(lstcal)/len(lstcal))
    maxss= max(lstcal)+sty
    minss= min(lstcal)-sty
    utime.sleep_us(200000)
    buzzer.freq(600)
    buzzer.duty_u16(1000)
    utime.sleep_us(500000)
    buzzer.duty_u16(0)
    tm.show("ECAL")
    utime.sleep_us(1000000)
    while True:
      impuls.value(1)
      utime.sleep_us(plen)
      impuls.value(0)
      utime.sleep_us(wper)
      vadc=rd.read_u16()
      if (vadc <= minss) or (vadc>=maxss):
        buzzer.duty_u16(1000)
        utime.sleep_us(frq*1000)
      else:
        utime.sleep_us(frq*1000)
      buzzer.duty_u16(0)
      dif=minss-vadc
      if dif<=0:
        dif=0
      tm.number(round(dif/100))
```

For using the application, download the file **main.py** and upload into the root of Raspberry Pi Pico. Also you will need to do the same with **tm1637.py** file, witch represents the library for TM1632 four digit LED display.



Some explanations about operating:

After powering up, the app is waiting for pressing a button. You ca press one of five buttons , witch have the following functions:

Button	Function
B1	Set the pulse length (plen). Pressing the button will scroll a list with preset values [microsec]. [50, 75, 100, 150, 200, 250, 300, 350, 400, 500]. Default value is 350. After pressing the button, the new value is displayed and setup for plen.
B2	Set the waiting perioad after the pulse is off (wper). Pressing the button will scroll a list with preset values[microsec]. [20, 30, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300]. Default value is 100. After pressing the button, the new value is displayed and setup for wper.
B3	Set the sensitivity (sty).). Pressing the button will scroll a list with preset values. [0, 50, 100, 150, 200, 250, 300, 350, 400, 500, 700]. Default value is 150. After pressing the button, the new value is displayed and setup for sty.
B4	Set the pulse frequency (frq).). Pressing the button will scroll a list with preset values. [20, 30, 40, 50, 60, 70, 80, 90, 100, 120, 140, 160]. Default value is 50. After pressing the button, the new value is displayed and setup for frq.
B5	This button will start metal detection operation.
B6	Reset the system

After setting the parameters (using B1, B2, B3, B4 buttons, if needed), pressing the B5 button will start the metal detection operation. For the beginning, a calibration is needed, that means a pulse is generated. After pulse is over, a perioad of time is waiting and then the adc input of microcontroler will read the voltage value from the OpAmp. This value is stored inside a list. This process is repeated six times, at the end we will have a list with six values. The application will calculate the average value of the list, also the min and max values and the calibration process is done.

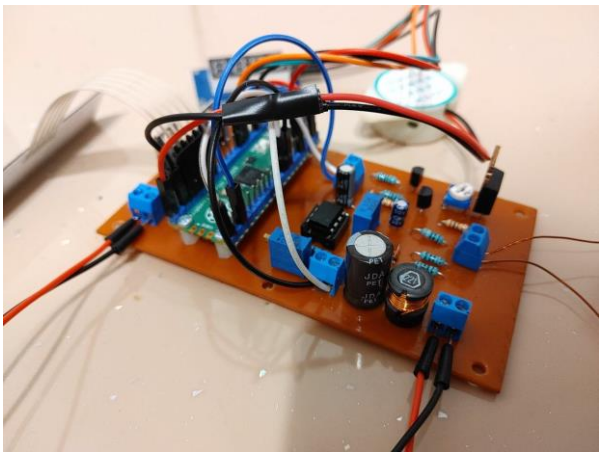
Warning ! The coil musk be kept away from any metalic surface or metalic component during calibration process.

After calibration process is finished, the application enter in a closed loop, where metal detection is made. As in calibration process, a pulse is generated, a time interval is waiting, then the adc input is read and the value is compared with min and max values (corrected with sty value) found in calibration process. If value is out of the range, this means a metalic target is nearby. The buzzer will transmitt a sound and also the display will show a value that represents the difference between the adc input value and min value. This cycle is continuously.

5. Construction hints.

It is very important to power the system with exactly 14v dc. This becose the operational amplifier has a major destruction risk for supply voltages greater then 15v dc. On the other side, higher power voltage means more powerfull magnetic field generated by search coil and a bigger sensitivity of detection process. I have found that 14v it is a good compromise. For field operation, I will power the device with a pack of 5 pieces Li-Ion batteries type 18650, followed by a voltage regulator with LM317. Also I will need a charging system for this batteries. For testing purposes it was used a stationary supply.

The Raspberry Pi Pico has to be soldered on the PCB directly. Unfortunately, my prototype PCB has a dimensional error so I couldn't soldered the Pico Board and was necessary to make the assembly with screws.



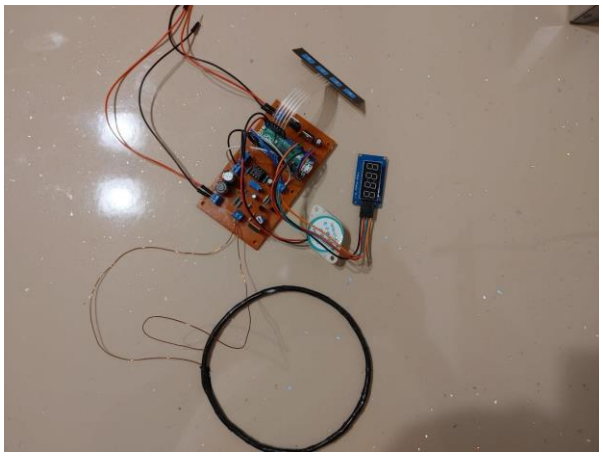
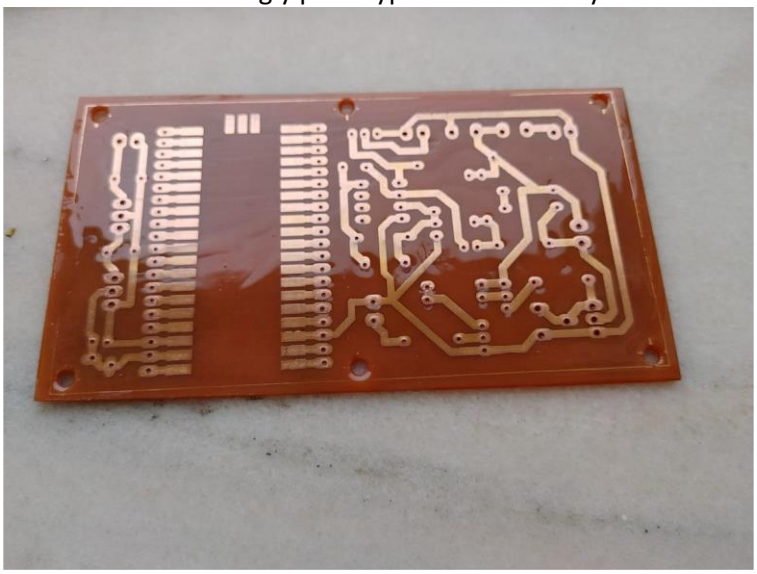


Photo: ugly prototype board asembly



PCB work

The search coil is very important, you can change the diameter of the form, the diameter of the wire and number of turns, depending of your needs, but all this have to be experimented. It is important that the resistence of the coil not to be less then 1 ohm. Also, smaller form diameter for the coil means better detection for smaller targets.

Operational amplifier has an important role regarding total sensitivity of the system. It is very important that this component to have a good quality and a very low noise. TL072 will meet this requirements. Replaceing the OpAmp with any other general purpose devices will decrease the sensitivity of the detector.

Proto inside Al enclosure – front view

Proto inside Al enclosure – rear view

6. Results

Reasults are almost good with respect to simplity of the design. With a 14V power supply and the serch coil described in schematic file, I have obtain some good detection distances:

- a piece of pcb 10x5 cm at 35cm
- a metal box 12x12cm at 40 cm
- a zippo lighter at 30 cm
- 1 euro coin at 20 cm
- smaller coins at 10-15 cm
- screw M4 X 40 at 15cm
- smaller screw M3 x 20 at 8 cm

7. Future work

There are a lot of improvements that can be done, but I hope to have some free time and to be focused on the followings:

- adding discrimination capabilities to the software
- adding a power supply for field operation
- new pcb design with reduced dimensions
- a final assembly

Daniel Mihai
daniel.v.mihai@gmail.com

Version 2
03/04/2021