

《电子系统综合设计与制作》中期报告

——智能磁阻式电磁炮台

江玮陶 2023010631

张光宇 2023010629

陈冠嘉 2023010503

2024 年 9 月 21 日

目录

1 项目背景	1
2 项目简介	2
3 子模块设计与实现	2
3.1 基于ZVS的升压电路	2
3.2 多级线圈驱动电路	2
3.3 瞄准系统	5
4 系统全貌	6
5 总结与展望	6
A 软件源码清单	6
A.1 MEGA主控代码	6
A.2 K210端代码	17
A.3 ESP8266/ESP32端代码	19
A.4 python仿真代码	27
B 相关链接	28
C 参考文献	28

1 项目背景

电磁炮是指使用电磁力发射炮弹的新型发射装置，其原理的提出已很久远，至少有近百年历史。随着脉冲电源技术的成熟和电脑仿真技术的出现，电磁炮技术自上世纪70年代起开始出现突破性进展，在学术和军事领域展现重要的潜在应用前景。

电磁炮从原理上分为线圈炮和轨道炮，前者又可以按照磁力的来源分为磁阻炮和感应炮两种。磁阻炮的加速原理类似级联的电磁铁，通过精巧的控制各级线圈的通电时机达到“接力”加速弹丸的目的，由于其结构简单、可拓展性强且不依赖急剧变化的电流，本项目采取多级磁阻作为炮体部分。

2 项目简介

本项目具有如下功能：

- 基于半桥拓扑的线圈驱动功能：采用半桥电路驱动线圈，实现高效能量传输和能量回收，提高电磁炮的效率和性能。
- 基于Arduino Mega、舵机云台的瞄准功能：利用Arduino Mega控制舵机云台，实现目标的自动瞄准和跟踪。
- 基于部署在K210端侧计算平台的YOL02模型的人体识别功能：利用K210强大的计算能力，部署YOL02模型进行人体识别，并返回目标位置和大小信息。

3 子模块设计与实现

3.1 基于ZVS的升压电路

为了给电磁炮提供足够的能量，本项目采用ZVS充电器电路进行线圈充电。该电路利用变压器和功率开关，将低压直流电转换为高压直流电，并为线圈提供能量。为了提高电路的可靠性和安全性，本项目采用带隔离控制的升压器，避免高压对控制电路的影响。为了引入单片机的控制，我们采用一个GPIO引脚通过光耦隔离控制一只场管下拉ZVS电路低电平的方式，实现了电路的控制。电路的工作原理和实物图如图 2 所示。

3.2 多级线圈驱动电路

为了实现电磁炮的线圈驱动，本项目采用半桥拓扑的线圈驱动电路。该电路利用两只场效应管和两只二极管，实现了对线圈的高效能量传输和能量回收。

根据基本电磁学知识不难知道，对于一个内外径分别为 r_1 和 r_2 的螺线管，其轴线上磁场强度 H 与电流 I 的关系为

$$H(z) = \frac{NI}{2LR} \left[(z+a) \ln \left(\frac{r_2 + \sqrt{(z+a)^2 + r_2^2}}{r_1 + \sqrt{(z+a)^2 + r_1^2}} \right) - (z-a) \ln \left(\frac{r_2 + \sqrt{(z-a)^2 + r_2^2}}{r_1 + \sqrt{(z-a)^2 + r_1^2}} \right) \right] \quad (1)$$

从而根据虚功原理，不难求得子弹的受力大小为

$$F = \frac{\partial W}{\partial z} = \frac{\partial \iiint_V \omega dV}{\partial z} \approx \frac{1}{2} \pi R^2 B_m (H_{top} - H_{bottom}) \quad (2)$$

考虑到本次使作炮弹的A3钢的饱和磁感应强度为1.0T量级，采用python进行数值模拟，得到了炮弹在磁场中的受力曲线，如图 3 所示。

可以观察到，只要让炮弹“卡”在运动的磁力波第一个波峰之后处，就可以直接利用时序控制的方式实现炮弹的加速。此时的磁场力与炮弹位置负相关，形成了相对稳定的负反馈，因此使用有一定误差的线圈驱动电路也可以实现较好的加速效果。

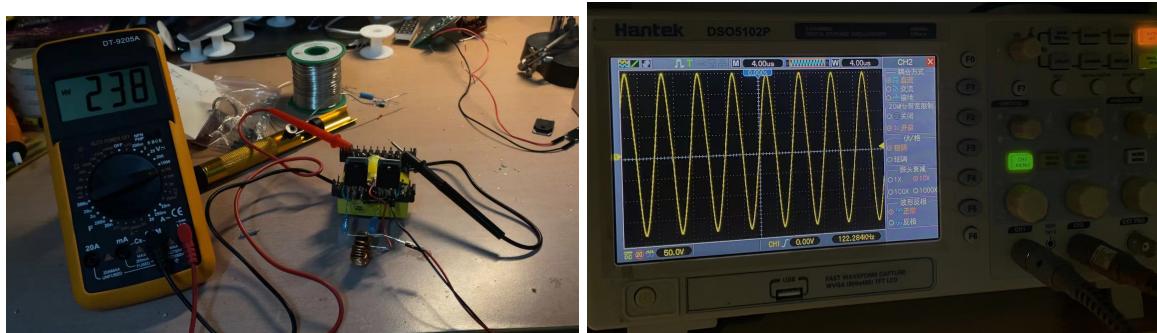
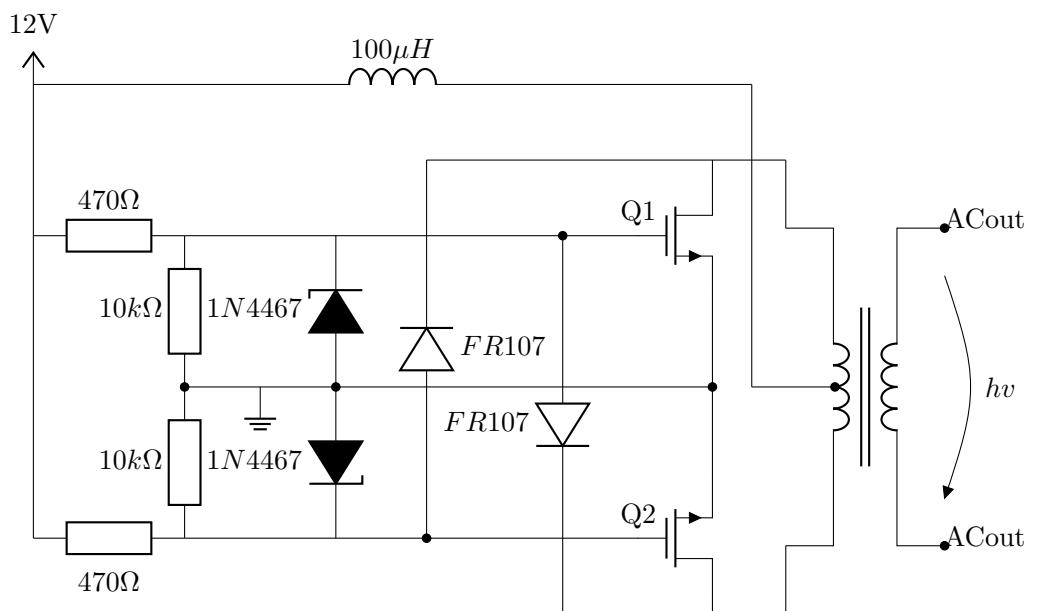


图 1: 基于ZVS的升压电路-理论验证

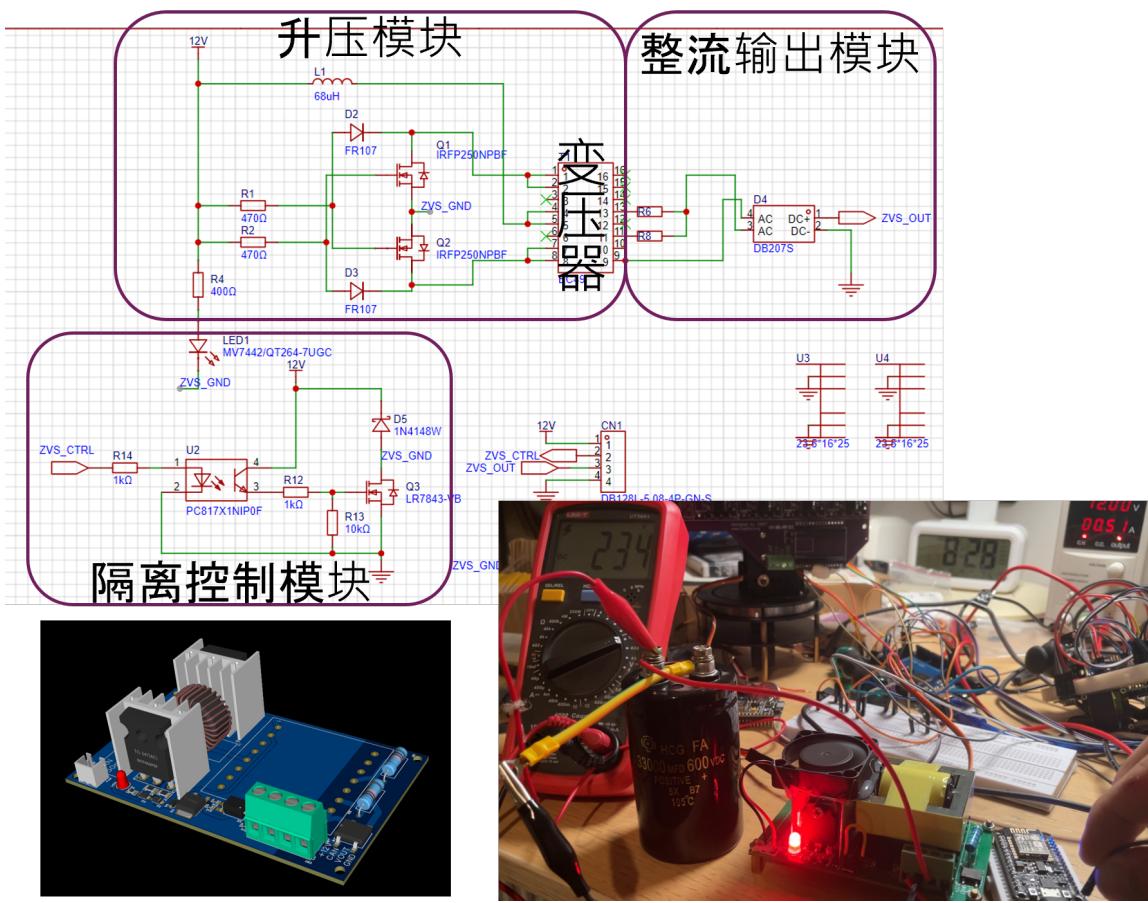


图 2: 基于ZVS的升压电路-实物图

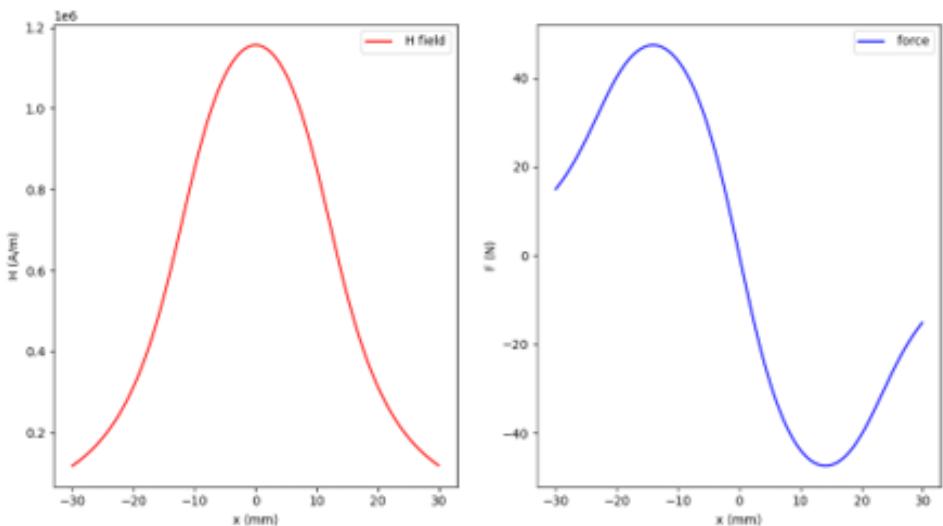


图 3: 线圈磁场力曲线

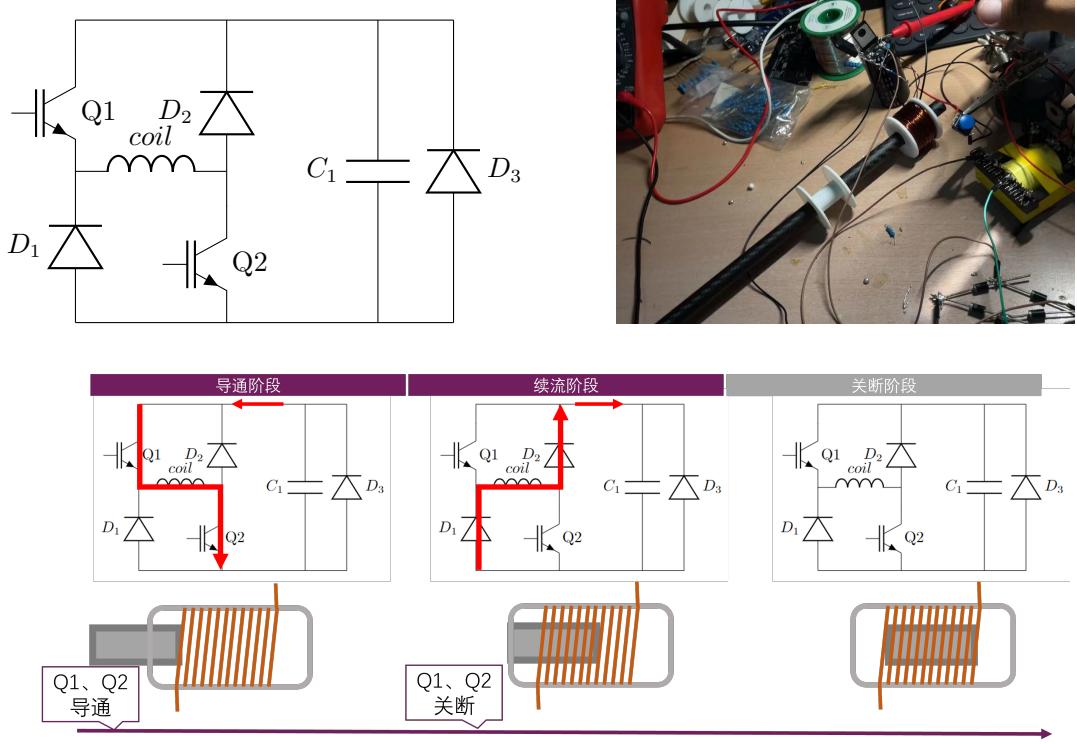


图 4: 半桥IGBT驱动电路示意及单级测试

我们设计半桥拓扑的线圈驱动电路，如图 4 所示。

完成单级测试后，打样PCB得到了六级线圈的控制板，如图 5 所示。

3.3 瞄准系统

为了实现电磁炮的瞄准功能，本项目采用Arduino Mega控制舵机云台，实现了目标的自动瞄准和跟踪。人体识别基于部署在K210端侧计算平台的YOLO2模型进行人体识别，并返回目标位置和大小信息。

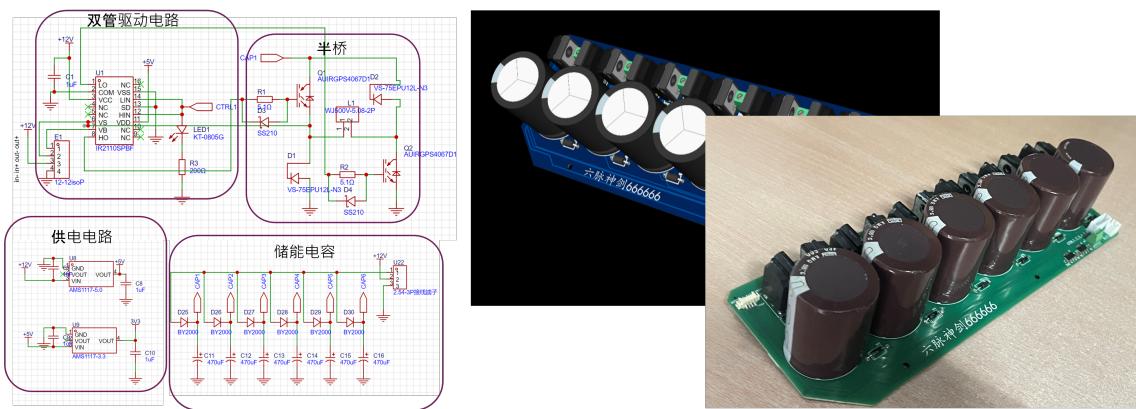


图 5: 线圈驱动电路PCB

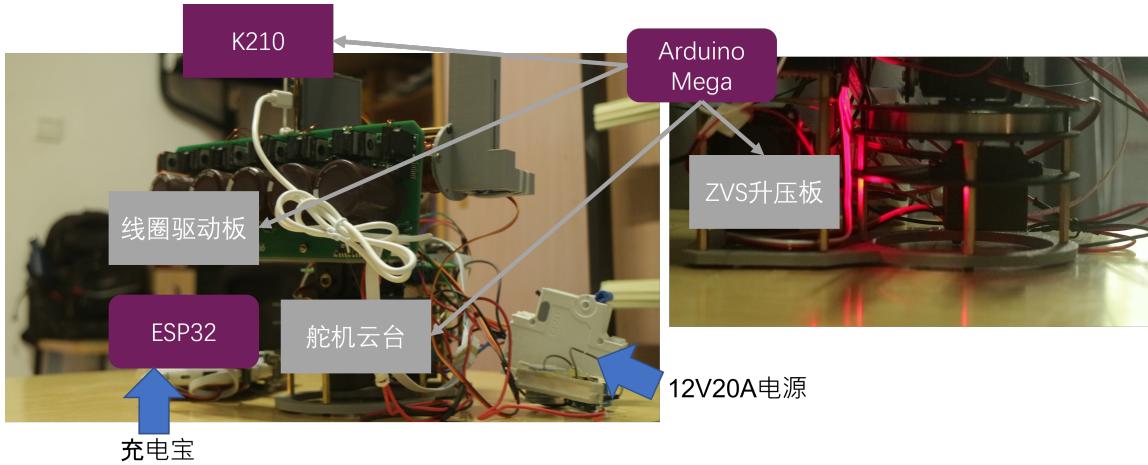


图 6: 系统全貌

4 系统全貌

系统组装后的全貌如图 6 所示。本项目的硬件架构主要包括 Arduino Mega、舵机云台、线圈驱动板、ESP8266、K210、ESP32、OLED 屏幕、按钮、摇杆模块、充电宝和 12V20A 电源等。Arduino Mega 作为主控单元，负责接收传感器数据、控制舵机云台和线圈驱动板，并与 K210 和 ESP32 进行通信。舵机云台用于瞄准目标，线圈驱动板用于驱动线圈，ESP8266 用于实现遥控功能，K210 用于人体识别，ESP32 用于连接 OLED 屏幕和按钮，OLED 屏幕用于显示系统状态，按钮和摇杆模块用于用户交互，充电宝和 12V20A 电源用于为系统供电。

5 总结与展望

本项目实现了智能电磁炮的基本功能，在电路、功能设计等方面得到了较好的效果。但本项目仍有较多的改进空间，横向和纵向上都有大量可供我们继续开发的方向。

横向还可以探索外设传感器和用户界面，例如超声波传感器、红外传感器、触摸屏等，以丰富电磁炮的功能和应用场景。

纵向上则可以探索线圈的最优几何构型和电路拓扑，例如多线圈结构、多相驱动电路等，从而提高电磁炮的性能和效率。此外，探索更高效的充电器设计，可以提高电磁炮的续航能力。

A 软件源码清单

A.1 MEGA 主控代码

Listing 1: main.c

```

1  /**
2   * @file main.cpp
3   * @author WeitaoJiang cbdt.tophu.org
4   * @brief

```

```

5      * @version 0.1
6      * @date 2024-08-10
7
8      * @copyright Copyright (c) 2024
9
10     */
11
12 #include <Arduino.h>
13 #include <Servo.h>
14 #include <SPI.h>
15 #include <RF24.h>
16 #include <nRF24L01.h>
17 #define IDLE_MODE 2
18 #define RC_MODE 0
19 #define AIM_MODE 1
20 #define DEBUG_MODE -1
21 #define K210 Serial2
22 #define ESP8266 Serial1
23
24 bool with_K210 = true;
25 bool Use_RC = true;
26
27 #define LASER_PIN 22
28 #define CHARGE_ENA_PIN 37
29 #define CHARGE_VOLTAGE_PIN A0
30
31 #define ANALOG_TO_VOLTAGE_MULTIPLIER (300 / 800.0)
32 typedef int switchmode_t;
33 switchmode_t mode = IDLE_MODE;
34
35 const int STAGE_CTRL_PINS[] = {38, 40, 42, 44, 46, 48};
36 const int INIT_PIN = 32;
37 const float aim_point_relative_x = 0.45;
38 const float aim_point_relative_y = 0.5;
39
40 Servo waist_servo, tilt_servo, load_servo;
41
42 // servo.write(servo.read() + Kp * err + Ki * int + Kd * diff);
43 /*PID const configurations*/
44 // waist
45 const float Kp_waist = 0.3;

```

```

46     const float Ki_waist = 0.0;
47     const float Kd_waist = 0.05;
48     // tilt
49     const float Kp_tilt = 0.4;
50     const float Ki_tilt = 0.0;
51     const float Kd_tilt = 0.1;
52
53     float int_waist = 0;
54     float diff_waist = 0;
55     float err_waist = 0;
56
57     float int_tilt = 0;
58     float diff_tilt = 0;
59     float err_tilt = 0;
60
61     // servo angle ctrl
62     int load_angle = 100;
63     int load_rest_angle = 10;
64
65     struct Cmd
66     {
67         switchmode_t mode;
68         int joy_x;
69         int joy_y;
70         int joy_btn_pressed;
71     } cmd;
72     /*PID functions*/
73     float PID_waist(float err_)
74     {
75
76         int_waist += err_;
77         diff_waist = err_ - err_waist;
78         err_waist = err_;
79         float ret = (Kp_waist * err_waist + Ki_waist * int_waist +
80                     Kd_waist * diff_waist) * 90.0;
81         if (ret > 10)
82             return 10;
83         if (ret < -10)
84             return -10;
85         return ret;
86     }

```

```

86     float PID_tilt(float err_)
87     {
88
89         int_tilt += err_;
90         diff_tilt = err_ - err_tilt;
91         err_tilt = err_;
92         return (Kp_tilt * err_tilt + Ki_tilt * int_tilt + Kd_tilt *
93                 diff_tilt) * 80.0;
94     }
95
96     void aim_target_1_itr(int target_x, int target_y, int
97                           canvas_width, int canvas_height)
98     {
99
100        int current_waist_angle = waist_servo.read();
101        int current_tilt_angle = tilt_servo.read();
102
103        float now_err_waist = -(target_x - canvas_width *
104                               aim_point_relative_x) / (float)canvas_width;
105        float now_err_tilt = -(target_y - canvas_height *
106                               aim_point_relative_y) / (float)canvas_height;
107
108        int delta_waist = PID_waist(now_err_waist);
109        int delta_tilt = PID_tilt(now_err_tilt);
110
111        waist_servo.write(current_waist_angle + delta_waist);
112        tilt_servo.write(current_tilt_angle + delta_tilt);
113        delay(20);
114        waist_servo.write(waist_servo.read());
115        tilt_servo.write(tilt_servo.read()); // protections
116    }
117
118    /*fire related funcs*/
119
120    int VSETlo = 335;
121    int VSEThi = VSETlo + 3; // voltage set points, sadly our zvs
122                            // cant pid
123
124    int stage_delay_us_arr[] = {0, 0, 0, 0, 0, 0};
125                            // delay after stage
126    long int stage_hold_us_arr[] = {1750, 800, 600, 420, 100, 0};
127                            // hold time

```

```

120
121     void fire()
122     {
123         digitalWrite(CHARGE_ENA_PIN, LOW);
124         load_servo.write(load_rest_angle);
125         delay(200);
126         load_servo.write(load_angle);
127         delay(200);
128         for (int i = 0; i < 6; i++)
129         {
130             if (stage_hold_us_arr[i] == 0)
131                 continue;
132             delayMicroseconds(stage_delay_us_arr[i]);
133             digitalWrite(STAGE_CTRL_PINS[i], HIGH);
134             delayMicroseconds(stage_hold_us_arr[i]);
135             digitalWrite(STAGE_CTRL_PINS[i], LOW);
136         }
137         load_servo.write(load_rest_angle);
138     }
139
140     int get_voltage()
141     {
142         return analogRead(CHARGE_VOLTAGE_PIN) *
143             ANALOG_TO_VOLTAGE_MULTIPLIER;
144     }
145
146     void update_charger()
147     {
148         // if (Use_RC)
149         // {
150         //     ESP8266.print("Voltage:");
151         //     ESP8266.print(get_voltage());
152         //     ESP8266.println(",Charging:1,Active:1");
153         // }
154         if (get_voltage() < VSETlo)
155         {
156             digitalWrite(CHARGE_ENA_PIN, HIGH);
157         }
158         else if (get_voltage() > VSEThi)
159         {
160             digitalWrite(CHARGE_ENA_PIN, LOW);

```

```

160     }
161     Serial.print("charging:");
162     Serial.println(get_voltage());
163     delay(100);
164 }
165 /*ctrl related funcs*/
166
167 float JOY_X_TO_WAIST = 10.0 / 4096.0;
168 float JOY_Y_TO_TILT = 5.0 / 4096.0;
169 int read_human_sensor()
170 {
171     return 1;
172 }
173
174 void alarm()
175 {
176     // Serial.println("Human detected!");
177     return; // todo: implement this function
178 }
179
180 /* MAIN */
181 void setup()
182 {
183     // put your setup code here, to run once:
184
185     Serial.begin(9600);
186     Serial.println("Hello, world!");
187     delay(1000);
188     if (with_K210)
189         K210.begin(9600);
190     if (Use_RC)
191         ESP8266.begin(9600);
192     waist_servo.attach(7);
193     waist_servo.write(155);
194     tilt_servo.attach(3);
195     tilt_servo.write(92);
196     load_servo.attach(2);
197
198     load_servo.write(load_rest_angle);
199     delay(1000);
200     pinMode(LASER_PIN, OUTPUT);

```

```

201     pinMode(CHARGE_ENA_PIN, OUTPUT);
202     pinMode(CHARGE_VOLTAGE_PIN, INPUT);
203     for (int i = 0; i < 6; i++)
204     {
205         pinMode(STAGE_CTRL_PINS[i], OUTPUT);
206     }
207     pinMode(INIT_PIN, OUTPUT);
208     delay(1000);
209     Serial.println("init...");
210     delay(1000);
211     digitalWrite(INIT_PIN, HIGH);
212     delay(1000);
213     digitalWrite(INIT_PIN, LOW);
214 }
215
216 void loop()
217 {
218     update_charger();
219     if (Use_RC && ESP8266.available())
220     {
221         String command = ESP8266.readStringUntil('\n');
222         Serial.print("ESP says:");
223         Serial.println(command);
224         cmd.joy_btn_pressed = 0;
225         sscanf(command.c_str(), "Mode:%d, JoyX:%d, JoyY:%d,
226                 JoyBtnPressed:%d", &cmd.mode, &cmd.joy_x, &cmd.joy_y, &
227                 cmd.joy_btn_pressed);
228     }
229     if (Use_RC == false)
230     {
231         cmd.mode = DEBUG_MODE;
232     }
233     switch (cmd.mode)
234     {
235
236     case DEBUG_MODE:
237
238         delay(200);
239         if (Serial.available())
240         {
241             String command = Serial.readString();

```

```

240     if (command.indexOf("fire") != -1)
241     {
242         Serial.println("got:fire!");
243         fire();
244     }
245 }
246 break;
247 case AIM_MODE:
248     if (true)
249     {
250         int times_not_found = 0;
251         alarm();
252         digitalWrite(LASER_PIN, HIGH);
253         while (read_human_sensor())
254         {
255             if (Use_RC && ESP8266.available())
256             {
257                 String command = ESP8266.readStringUntil('\n');
258                 Serial.print("ESP says:");
259                 Serial.println(command);
260                 cmd. joy_btn_pressed = 0;
261                 sscanf(command.c_str(), "Mode:%d, JoyX:%d, JoyY:%d,
262                     JoyBtnPressed:%d", &cmd.mode, &cmd. joy_x, &cmd.
263                     joy_y, &cmd. joy_btn_pressed);
264             if (cmd.mode == RC_MODE)
265             {
266                 break;
267             }
268         }
269         /*charge and check*/
270         update_charger();
271         if (!with_K210)
272             delay(1000);
273         /*read from K210 and aim.*/
274         // K210.write("get");
275         if (with_K210 && K210.available())
276         {
277             String text = String(K210.readStringUntil('\n'));
278             delay(10);
279             Serial.println(text);

```

```

279     if (text.indexOf("target not found") != -1)
280     {
281         times_not_found++;
282         if (times_not_found > 10)
283         {
284
285             Serial.println("Target not found for 10 times, stop aiming.");
286             /*set angle to (90,30)*/
287
288             while (tilt_servo.read() > 90)
289             {
290                 tilt_servo.write(tilt_servo.read() - 1);
291                 delay(40);
292             }
293             while (tilt_servo.read() < 90)
294             {
295                 tilt_servo.write(tilt_servo.read() + 1);
296                 delay(40);
297             }
298             while (waist_servo.read() > 155)
299             {
300                 waist_servo.write(waist_servo.read() - 1);
301                 delay(40);
302             }
303             while (waist_servo.read() < 155)
304             {
305                 waist_servo.write(waist_servo.read() + 1);
306                 delay(40);
307             }
308             /*set pid vars to 0*/
309             int_waist = 0;
310             diff_waist = 0;
311             err_waist = 0;
312             int_tilt = 0;
313             diff_tilt = 0;
314             err_tilt = 0;
315
316             int tmp_var_waist = -20;
317             int tmp_var_tilt = -10;
318             /*start searching for target

```

```

319         waist: 140~160 160-(20~0)
320         tilt: 90~120 120-(30~0)
321     */
322     // (140,160)
323     while (true)
324     {
325         if (Use_RC && ESP8266.available())
326         {
327             String command = ESP8266.readStringUntil('\n'
328                 );
329             Serial.print("ESP says:");
330             Serial.println(command);
331             cmd.joy_btn_pressed = 0;
332             cmd.joy_btn_pressed=0;
333             sscanf(command.c_str(), "Mode:%d, JoyX:%d,
334                 JoyY:%d, JoyBtnPressed:%d", &cmd.mode, &
335                 cmd.joy_x, &cmd.joy_y, &cmd.
336                 joy_btn_pressed);
337             if (cmd.joy_btn_pressed)
338             {
339                 if (get_voltage() > VSETlo)
340                 {
341                     fire();
342                 }
343             }
344         }
345         update_charger();
346         text = String(K210.readStringUntil('\n'));
347         Serial.println("searching for target, got:" + 
348             text);
349         if (text.indexOf("aim:") != -1)
350         {
351             int target_x, target_y, canvas_width,
352                 canvas_height, target_size;
353             sscanf(text.c_str(), "aim:@%d@%d@canvas:@%d@
354                 %d@size:@%d", &target_x, &target_y, &
355                 canvas_width, &canvas_height, &

```

```

            target_size);
352         aim_target_1_itr(target_x, target_y,
353             canvas_width, canvas_height);
354         times_not_found = 0;
355         break;
356     }
357     tilt_servo.write(100 - abs(tmp_var_tilt));
358     waist_servo.write(175 - abs(tmp_var_waist));
359     tmp_var_waist += 1;
360     tmp_var_tilt += 1;
361     if (tmp_var_waist > 20)
362     {
363         tmp_var_waist = -20;
364     }
365     if (tmp_var_tilt > 10)
366     {
367         tmp_var_tilt = -10;
368     }
369     delay(100);
370     continue;
371 }
372 }
373 if (text.indexOf("aim:") != -1)
374 {
375     int target_x, target_y, canvas_width,
376         canvas_height, target_size;
377     sscanf(text.c_str(), "aim:%d%dcanvas:%d%d"
378             "size:%d", &target_x, &target_y, &canvas_width,
379             &canvas_height, &target_size);
380     aim_target_1_itr(target_x, target_y, canvas_width,
381                     canvas_height);
382     times_not_found = 0;
383 }
384 break;
385 case RC_MODE:
386     if (cmd.joy_btn_pressed)

```

```

387     {
388         if (get_voltage() > VSETlo)
389         {
390             fire();
391         }
392     }
393
394     break;
395 }
396 }
```

A.2 K210端代码

Listing 2: main.py

```

1 from machine import UART,Timer
2 from fpioa_manager import fm
3 import sensor, image, time, lcd
4 from maix import KPU
5 import gc
6
7
8
9 fm.register(6,fm.fpioa.UART1_RX,force=True)
10 fm.register(7,fm.fpioa.UART1_TX,force=True)
11
12 uart1 = UART(UART.UART1, 9600)
13 uart1.write('Hello\u2022From\u2022K210!')
14 lcd.init()
15
16 sensor.reset()
17 sensor.set_vflip(1)
18 sensor.set_hmirror(1)
19
20 sensor.set_pixformat(sensor.RGB565) # Set pixel format to RGB565
21 # (or GRayscale)
22 sensor.set_framesize(sensor.QVGA)    # Set frame size to QVGA (320
23 # x240)
24 sensor.skip_frames(time = 1000)       # Wait for settings take
25 # effect.
```

```

23  clock = time.clock()                      # Create a clock object to
     track the FPS.
24  od_img = image.Image(size=(320,256))
25
26
27  anchor_body_detect = (0.0978, 0.1758, 0.1842, 0.3834, 0.3532,
28          0.5982, 0.4855, 1.1146, 0.8869,
29          1.6407, 1.2388, 3.4157, 2.0942, 2.1114,
30          2.7138, 5.0008, 6.0293, 6.4540)
31
32
33  body_kpu = KPU()
34
35  body_kpu.load_kmodel("/sd/uint8_person_detect_v1_old.kmodel")
36
37  body_kpu.init_yolo2(anchor_body_detect, anchor_num=9, img_w=320,
38          img_h=240, net_w=320 ,
39          net_h=256 ,layer_w=10 ,layer_h=8, threshold
40          =0.7, nms_value=0.2, classes=1)
41
42  aim_x = -1
43  aim_y = -1
44  body_size = -1
45  while True:
46      gc.collect()
47      clock.tick()                      # Update the FPS clock.
48      img = sensor.snapshot()
49      a = od_img.draw_image(img, 0,0)
50      od_img.pix_to_ai()
51
52      body_kpu.run_with_output(od_img)
53      body_boxes = body_kpu.regionlayer_yolo2()
54      if len(body_boxes) > 0:
55
56          for l in body_boxes :
57              aim_x = int(l[0]+l[2]/2)
58              aim_y = int(l[1]+l[3]*0.5)
59              body_size = l[2]*l[3]
60              a = img.draw_rectangle(l[0],l[1],l[2],l[3], color
61              =(255, 0, 0),thickness=2)
62              img.draw_cross(int(l[0]+l[2]/2),int(l[1]+l[3]*0.5),
63              color=(255,255,255),size=10,thickness=1)
64
65      else:

```

```

57         aim_x = -1
58         aim_y = -1
59         body_size = -1
60     if aim_x >= 0 and aim_y >= 0:
61         uart1.write("aim:"+str(aim_x)+"_"+str(aim_y)+"_canvas:"
62                         +"_"+str(od_img.width())+"_"+str(od_img.height())+"_"+"size:_"
63                         +"_"+str(body_size)+"\n")
64     else:
65         uart1.write("target not found\n")
66     fps = clock.fps()
67     a = img.drawString(0, 0, "%2.1ffps" %(fps), color=(0, 60,
68                       128), scale=2.0)
69     lcd.display(img)
70     body_kpu.deinit()

```

A.3 ESP8266/ESP32端代码

Listing 3: esp8266.ino

```

1 #include<ESP8266WiFi.h>
2 #include <WiFiClient.h>
3 #include <WiFiServer.h>
4 #include<Arduino.h>
5 #include<HardwareSerial.h>
6 #include<stdio.h>
7 #include<string.h>
8
9 #include<wl_definitions.h>
10
11 const char* ssid="esp8266";
12 const char* password="wtgg6666";
13 const int port = 8080;
14
15 WiFiServer server(port);
16
17 void setup() {
18     Serial.begin(9600);
19     WiFi.softAP(ssid,password);
20     //WiFi.begin(ssid,password);
21     Serial.print("\nAccess Point:");
22     Serial.println(ssid);

```

```

23     Serial.print("IP address: ");
24     Serial.println(WiFi.softAPIP());
25
26     //while ( WiFi.status() != WL_CONNECTED) {
27     //  delay(1000);
28     //  Serial.print(".");
29     //}
30
31     //Serial.println("WiFi connected");
32     //Serial.println("IP address: ");
33     //Serial.println(WiFi.localIP());
34
35     server.begin();
36     Serial.println("Server started");
37
38 }
39 void loop() {
40     WiFiClient client = server.available();
41     if (client) {
42         //Serial.println("New client connected");
43         //delay(2000);
44         if(Serial.available()){
45             /*Serial.println("Voltage:");
46             delay(1000);
47             int voltage=Serial.read();
48             Serial.println("Charging?:");
49             delay(1000);
50             bool charging=Serial.read();
51             Serial.println("Active?:");
52             delay(1000);
53             bool active=Serial.read();
54             */
55             char* chpr=new char[100];
56             sprintf(chpr,Serial.readStringUntil('\n').c_str());
57             //sprintf(chpr,"Voltage:%d,Charging:%d,Active:%d",
58             //        voltage,charging,active);
59             client.println(chpr);
60             Serial.print(chpr);
61             delete chpr;
62         }
63         //client->server(>client)

```

```

63         //int mode ,joyx ,joyy ,joybtn;
64         //int flag=-1;
65         while (client.connected()) {
66             if (client.available()) {
67                 String line = client.readStringUntil('\n');
68                 //Serial.print("Received from client: ");
69                 Serial.println(line);
70                 //client.println(line);
71                 //flag=sscanf(line.c_str(),"Mode:%d,JoyX:%d,JoyY:%d,
72                 JoyBtnPressed:%d",mode,joyx,joyy,joybtn);
73             }
74             client.stop();
75             //Serial.println("Client disconnected");
76             //delay(1000);
77         }
78     }

```

Listing 4: ESP32.ino

```

1      #include <WiFi.h>
2      #include<Arduino.h>
3      #include<HardwareSerial.h>
4      #include<stdio.h>
5      #include<stdlib.h>
6      #include<string.h>
7      #include<U8g2lib.h>
8      U8G2_SSD1306_128X32_UNIVISION_F_HW_I2C u8g2(U8G2_R0, /* reset
   */ U8X8_PIN_NONE);
9
10     // #include<wl_definitions.h>
11
12     const char* ssid = "esp8266";
13     const char* password = "wtgg6666";
14     const char* host = "192.168.4.1";
15     const int port = 8080;
16     const int Rx=32,Ry=33,SW=25;
17     const int Btn=13;
18     const int R=12,G=14,B=27;
19     WiFiClient client;
20

```

```

21     void draw1(void) {
22         u8g2.setFont(u8g2_font_ncenB08_tr);
23         u8g2.drawStr(16, 16, "Electromagnetic");
24         u8g2.drawStr(40, 28, "Cannon");
25     }
26     void draw2(void) {
27         u8g2.setFont(u8g2_font_ncenB08_tr);
28         u8g2.drawStr(32, 20, "Connected!");
29     }
30     void draw3(void) {
31         u8g2.setFont(u8g2_font_ncenB08_tr);
32         u8g2.drawLine(20,32,68,32);
33         u8g2.drawLine(20,30,68,30);
34         u8g2.drawLine(20,32,20,30);
35         u8g2.drawLine(68,32,68,30);
36         u8g2.drawLine(22,30,22,28);
37         u8g2.drawLine(22,28,64,28);
38         u8g2.drawLine(64,30,64,28);
39         u8g2.drawLine(40,24,40,28);
40         u8g2.drawLine(48,24,48,28);
41         u8g2.drawLine(40,24,48,24);
42         u8g2.drawLine(40,18,42,26);
43         u8g2.drawLine(44,18,46,26);
44         u8g2.drawLine(30,18,71,8);
45         u8g2.drawLine(30,17,62,9);
46         u8g2.drawLine(30,19,62,11);
47         u8g2.drawLine(29,16,30,20);
48         u8g2.drawLine(34,15,34,19);
49         u8g2.drawLine(37,14,38,18);
50         u8g2.drawLine(41,13,42,17);
51         u8g2.drawLine(45,12,46,16);
52         u8g2.drawLine(49,11,50,15);
53         u8g2.drawLine(53,10,54,14);
54         u8g2.drawLine(57,9,58,13);
55         u8g2.drawLine(61,8,62,12);
56         u8g2.drawLine(88,4,90,6);
57         u8g2.drawLine(90,6,96,5);
58         u8g2.drawLine(96,5,99,2);
59         u8g2.drawLine(99,2,94,2);
60         u8g2.drawLine(94,2,88,4);
61         u8g2.drawLine(12,8,9,18);

```

```

62     u8g2.drawLine(9,18,13,16);
63     u8g2.drawLine(13,16,12,24);
64     u8g2.drawLine(12,24,15,14);
65     u8g2.drawLine(15,14,11,16);
66     u8g2.drawLine(11,16,12,8);
67     u8g2.drawStr(76,24,"Electrify!");
68 }
69
70 void setup() {
71     Serial.begin(9600);
72     WiFi.begin(ssid, password);
73     pinMode(SW, INPUT_PULLUP);
74     pinMode(Btn, INPUT);
75     pinMode(Rx, INPUT);
76     pinMode(Ry, INPUT);
77     pinMode(R, OUTPUT);
78     pinMode(G, OUTPUT);
79     pinMode(B, OUTPUT);
80     u8g2.begin();
81     u8g2.firstPage();
82     do {
83         draw3();
84     } while(u8g2.nextPage());
85     delay(3000);
86     u8g2.firstPage();
87     do {
88         draw1();
89     } while(u8g2.nextPage());
90     delay(2000);
91
92     while (WiFi.status() != WL_CONNECTED) {
93         delay(1000);
94         Serial.print(".");
95     }
96
97     Serial.println("WiFi connected");
98     Serial.println("IP address: ");
99     Serial.println(WiFi.localIP());
100 }
101
102 }
```

```

103     int connection=0;
104     int btn=0,btn_pre=0,btn_swch=0,btn_init=0;
105     void loop() {
106         if (!client.connect(host, port)) {
107             connection=0;
108             digitalWrite(R,LOW);
109             digitalWrite(G,LOW);
110             digitalWrite(B,LOW);
111             Serial.println("Connection to server failed");
112
113             u8g2.firstPage();
114             do {
115                 draw3();
116             } while(u8g2.nextPage());
117             delay(3000);
118
119             u8g2.firstPage();
120             do {
121                 draw1();
122             } while(u8g2.nextPage());
123             delay(2000);
124             return;
125         }
126
127         if(connection==0){
128             Serial.println("Connected to server");
129             u8g2.firstPage();
130             do {
131                 draw2();
132             } while(u8g2.nextPage());
133         }
134         connection=1;
135
136         int flag=-1,flag2=0;
137         int voltage=0,charging=0,active=0;
138         if (client.available()) {
139             String line = client.readStringUntil('\n');
140             //Serial.print("Received from server: ");
141             Serial.println(line);
142             flag=sscanf(line.c_str(),"Voltage:%d,Charging:%d,Active:%
d",&voltage,&charging,&active);

```

```

143 }
144 if(flag != -1) {
145     int progress ,progressX ,progressWidth ,progressHeight ;
146     u8g2.firstPage();
147     do {
148         u8g2.setFont(u8g2_font_ncenB08_tr);
149         u8g2.setCursor(36, 8);
150         if (charging == 0 && active == 1) {
151             u8g2.print("Active");
152         }
153         else if(charging == 1 && active == 0) {
154             u8g2.print("Charging");
155         }
156         else {
157             Serial.print("Error");
158             flag2=1;
159         }
160         if(flag2==0){
161             u8g2.setCursor(24, 16);
162             u8g2.print("Voltage:");
163
164             u8g2.setCursor(70, 16);
165             u8g2.print(voltage);
166             u8g2.print("V");
167
168             progress = (int)(voltage / 3.0);
169             progressX = 14;
170             progressWidth = 100;
171             progressHeight = 8;
172             u8g2.drawFrame(progressX, 22, progressWidth,
173                             progressHeight);
174
175             if (progress > 0) {
176                 u8g2.drawBox(progressX, 22, progress ,
177                             progressHeight);
178             }
179         } while(u8g2.nextPage());
180         flag=-1;
181         flag2=0;
182     }

```

```

182
183     //client.println("Hello from ESP32 Client");
184
185     int x=analogRead(Rx);
186     int y=analogRead(Ry);
187     int sw=1-digitalRead(SW);
188     btn=digitalRead(Btn);
189     btn_swch=abs(btn-btn_pre);
190     btn_init+=btn_swch;
191     btn_init%=4;
192     //Serial.println(btn);//
193     char* chtr=new char[100];
194     sprintf(chtr,"Mode:%d,JoyX:%d,JoyY:%d,JoyBtnPressed:%d",
195             btn_init/2,x,y,sw);
196     client.println(chtr);
197     //client.print("\n");
198     Serial.println(chtr);
199     //Serial.print("\n");
200     digitalWrite(R,LOW);
201     digitalWrite(G,LOW);
202     digitalWrite(B,LOW);
203     if(btn_init==0){
204         digitalWrite(G,HIGH);
205         delay(100);
206     }
207     else if(btn_init==2){
208         digitalWrite(B,HIGH);
209         delay(100);
210     }
211     else{
212         digitalWrite(R,HIGH);
213     }
214
215     //client.println("ESP32 Client received your message");
216
217     client.stop();
218     //Serial.println("Client disconnected");
219
220     btn_pre=btn;
221

```

```
222         delay(200);  
223     }
```

A.4 python仿真代码

Listing 5: force.py

```
1 import math  
2 from scipy.interpolate import interp1d  
3 import numpy as np  
4 import pandas as pd  
5  
6 from scipy.optimize import curve_fit  
7 import matplotlib.pyplot as plt  
8  
9 bullet_len_mm = 22  
10 bullet_rad_mm = 8/2  
11 class Coil:  
12     def __init__(self, location, length, N, innerRadius,  
13                  outerRadius):  
14         self.length = length  
15         self.innerRadius = innerRadius  
16         self.outerRadius = outerRadius  
17         self.N = N  
18         self.u0 = 4 * math.pi * 1e-7  
19     def HFieldCoil3(self, current, z):  
20         L=self.length  
21         a=L/2  
22         r2=self.outerRadius  
23         r1=self.innerRadius  
24         R=r2-r1  
25         def logarithm(pos):  
26             return pos * math.log((math.sqrt(pos ** 2 + r2 ** 2)  
27                         + r2) /  
28                         (math.sqrt(pos ** 2 + r1 **  
29                         2) + r1))  
30  
31         return ( self.N*current/(2*L*R) ) * (logarithm(z+a)-  
32             logarithm(z-a))  
33 coil = Coil(0, 23.4*1e-3, 200, 0.5*12.4*1e-3, 0.5*40*1e-3)  
34 current = 200
```

```

31 plt.figure(figsize=(10,10))
32 xx=[ pos for pos in np.arange(-30,30,0.1)]
33 yy=[ coil.HFieldCoil3( current, x*1e-3) for x in xx]
34 Fyy = []
35 for i in range (0,len(xx)):
36     Fyy.append((coil.HFieldCoil3( current, (xx[i]+bullet_len_mm
37                         /2)*1e-3)
38                         -coil.HFieldCoil3( current, (xx[i]-bullet_len_mm
39                         /2)*1e-3))
40                         *(bullet_rad_mm*1e-3)**2*math.pi)
41 plt.subplot(1,2,1)
42 plt.plot(xx, yy, color='red', label='H field')
43 plt.xlabel('x (mm)')
44 plt.ylabel('H (A/m)')
45 plt.legend()
46 plt.subplot(1,2,2)
47 plt.plot(xx, Fyy, color='blue', label='force')
48 plt.xlabel('x (mm)')
49 plt.ylabel('F (N)')
50 plt.legend()
51 plt.show()

```

B 相关链接

可点击 视频链接观看演示视频。

本项目开源，github仓库为此链接。

C 参考文献

1. 李小鹏, 徐征, 李立毅:《电磁线圈发射原理》, 北京: 兵器工业出版社, 2019.
2. 清华大学电力系高压技术专业编著.冲击大电流技术.北京: 科学出版社, 1978.139-141.
3. 王莹, 肖峰:《电炮原理》, 1993.
4. 李国林:《电子电路与系统基础》, 北京: 清华大学出版社, 2020