VDM-22 Sapfir

An RP-22 Sapir Radar Scope implementation for the ATmega 328P

# Overview of Vector Drawing Machine - 22

## Real World Operation

The RP-21/22 Sapfir was an analog radar system developed by the Soviet Union. Used primarily by the MiG-21 Bis, it can detect a fighter sized object from 20 km away flying at the same altitude. The radar antenna scans in a circular motion and emits a characteristic three tone signature on NATO RWRs, earning it the NATO nickname “Spin Scan”. It is capable of operating in three modes – Scan Mode, Chase Mode (locked on) and Fixed Beam (for guiding beam-riding ground attack rockets).

The interface between the RP-22 and the pilot comes in the form of a circular CRT situated to the lower right side of the MiG-21’s cockpit. The following pictures will depict the two air-to-air operation mode of the radar (Fixed Beam for Kh22 is for ground attack only, and will not be implemented in the current iteration):

1. B-Scope (Detection). 2. Chase Mode

A picture containing control panel

Description automatically generated

In B-Scope mode, the two horizontal line represents the “lock piper”. When the lock piper is over the bogey (the bird) and the TDC lock button is depressed, the RP-22 will scan the airspace designated by the piper at a high frequency and lock onto any target it sees.

When used alongside the SRZO-2 IFF interrogator/transponder, the RP-22 is capable of rudimentary IFF operation. When the IFF button is depressed on the radar control panel, the bird will turn into two parallel horizontal lines if the target is friendly and a single line if the target is hostile/unknown.

When the target is locked following the correct procedure, the RP-22 will enter the “chase mode”. Now instead of a top-down view (a B-scope), the Sapfir will display the C-Scope. The C-Scope displays the azimuth and deviation of the bogey from the centerline of our aircraft, with the two lines on the side representing the distance to target.

When the target is in range for the R3R SARH missile, a red lamp will illuminate with the wording “IN RANGE”. When the target is range, another red lamp “launch” will be lit, giving the pilot launch authorization when he/she presses the missile launch button.

## Gameplay

The gameplay will follow the perspective of a MiG-21/J-7 pilot at the end of GCI guidance. Upon game start, a bogey will appear on the very top edge of the scope (around 30 km). The initial delta V of the two aircrafts will be 800 kts (1481.6kph), which means the player will have about 40 seconds to maneuver the aircraft to lock up the bogey at 15 km. The player will now have to flip on a row of switches – Left Rack Power, Right Rack Power, Left Rack Launch Authorization and Right Rack Launch Authorization to prepare for the engagement.

Once the player obtains a lock, the interface will be switched to the C-Scope. The bogey has a random chance to start maneuvering and / or lock onto the player’s aircraft. The bogey’s radar status will be reflected by the MiG’s helpful soviet RWR. The RWR, following its real-life model, will consist of one sound unit and 4 LEDS representing the radar emission aspect to own aircraft. When the bogey’s radar is scanning the player’s aircraft, the RWR will beep with the LED flashing in the direction the bogey is coming from. If the bogey obtains a lock / launches a missile, the LED will turn solid, and the beep will turn into a continuous tone/very high frequency beep. If the bogey obtains a lock on the player, the player must hit the countermeasure dispense button to deploy chaff. For gameplay’s sake, the chaff deployed will have a 100% possibility of *confusing the bogey’s targeting system* thus ensuring the player’s survival.

Now if the player manages to evade the bogey’s missile and comes within launch distance (around 7km with both IN RANGE and LAUNCH lamps lit), the player will launch the R3R missile by pressing the missile launch button. If the player launches the missile before the distance closes within 2km, the bogey will be counted as shot down; otherwise, the player will be splashed by a capitalist AIM-9 sidewinder missile.

Upon finishing one game, the player will press the countermeasures dispense button to start the next round. The next round will multiply the delta V of the player’s aircraft and the bogey’s aircraft by 1.05, and the game will terminate to display score information on the vector scope when delta V reaches 1500 knots.

# Psudo-Code

#define C\_DAC\_DB0 23,OUTPUT  
#define C\_DAC\_DB1 24,OUTPUT  
#define C\_DAC\_DB2 25,OUTPUT  
#define C\_DAC\_DB3 26,OUTPUT  
#define C\_DAC\_RLDAC 5,OUTPUT  
#define C\_DAC\_DB4 6,OUTPUT  
#define C\_DAC\_DB5 11,OUTPUT  
#define C\_DAC\_DB6 12,OUTPUT  
#define C\_DAC\_DB7 13,OUTPUT  
#define C\_DAC\_RAB 14,OUTPUT  
#define C\_DAC\_RWR 16,OUTPUT  
  
#define INIT\_DIGITAL\_PINS \  
pinMode(C\_DAC\_DB0);\  
pinMode(C\_DAC\_DB1);\  
pinMode(C\_DAC\_DB2);\  
pinMode(C\_DAC\_DB3);\  
pinMode(C\_DAC\_DB4);\  
pinMode(C\_DAC\_DB5);\  
pinMode(C\_DAC\_DB6);\  
pinMode(C\_DAC\_DB7);\  
pinMode(C\_DAC\_RLDAC);\  
pinMode(C\_DAC\_RAB);\  
pinMode(C\_DAC\_RWR);\

#define C\_IN\_STICKX 27  
#define C\_IN\_STICKY 28  
  
#define INIT\_ANALOG\_PINS \  
Serial.begin(9600);  
  
#define getStickXVoltage() analogRead(C\_IN\_STICKX) \* (5.0 / 1023.0)  
#define getStickYVoltage() analogRead(C\_IN\_STICKY) \* (5.0 / 1023.0)

gameplay.c:  
enum State{Lock = 0,Unlocked = 1};  
  
draw.c:  
 Coordinate[] coordinateToDraw;  
void Draw\_\_draw();  
  
coordinate.c  
struct Coordinates  
{  
 short x;  
 short y;  
};  
void Coordinates\_\_set(Coordinate\* coordinate,const short x,const short y);  
short Coordinates\_\_getX(const Coordinate\* coordinate);  
short Coordinates\_\_getY(const Coordinate\* coordinate);

I have already set up the pin assignments and modes. The flow of the game will be based on the above functions defined in the headers. More code will be added throughout our designing process. It is impossible to cover the psudo-code of our entire program at this time, because our designs are more complicated.

Our code will be written in pure C with extensive usages on pointers and memory management to save resources.

# Hardware Description

Revised from the preliminary proposal, the entire game will be powered by one ATMega328P microcontroller. Instead of using the unreliable PWM output of the microcontroller itself, we will use an AD7302 dual 8-bit DAC to complete the DAC operation.

We will then use two pins from the ADC port for joystick input, 2 pins for launch / countermeasures dispense, 1 pin for MASTER ARM (Rail Power), 2 pins for RWR LED control and finally 1 pin for buzzer RWR feedback. Combined we will utilize 14 pins for I/O purposes. The current schematic state and block diagram is included below:

Diagram, schematic

Description automatically generated

Diagram

Description automatically generated

# Enclosure Design

Diagram

Description automatically generated

# Team Roles:

Damien Hu: Hardware/Testing Engineer

Yinhao Qian: Software Engineer

Shaoyu Pei: Enclosure/Testing Engineer