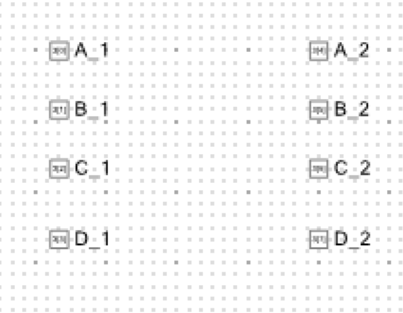
Basketball Bot

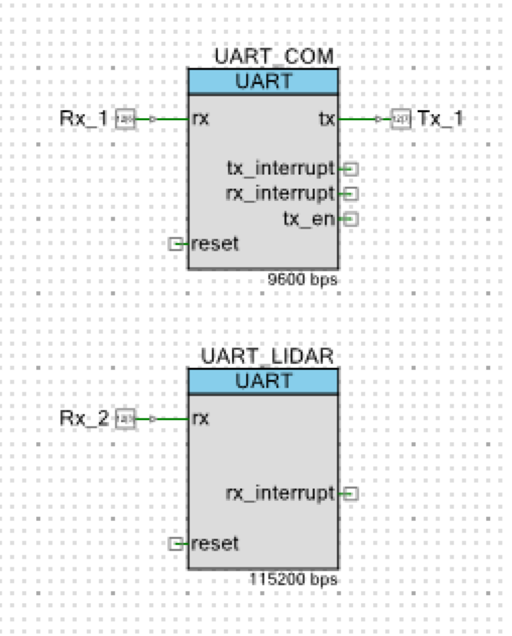
Colin Acton, Christine Cummings,
Darius Dastur, Miles Luhn

Top Design

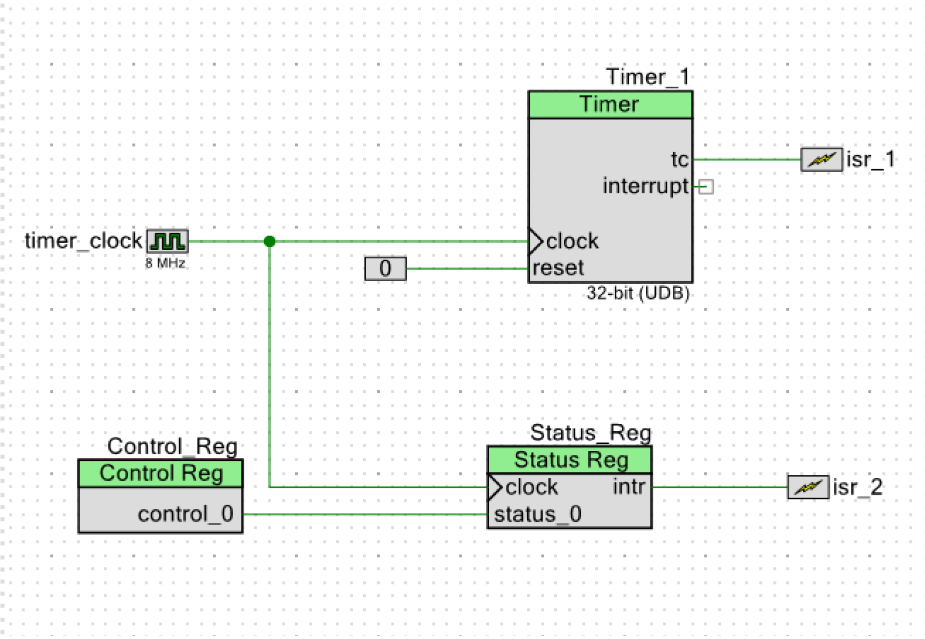


Stepper Motors

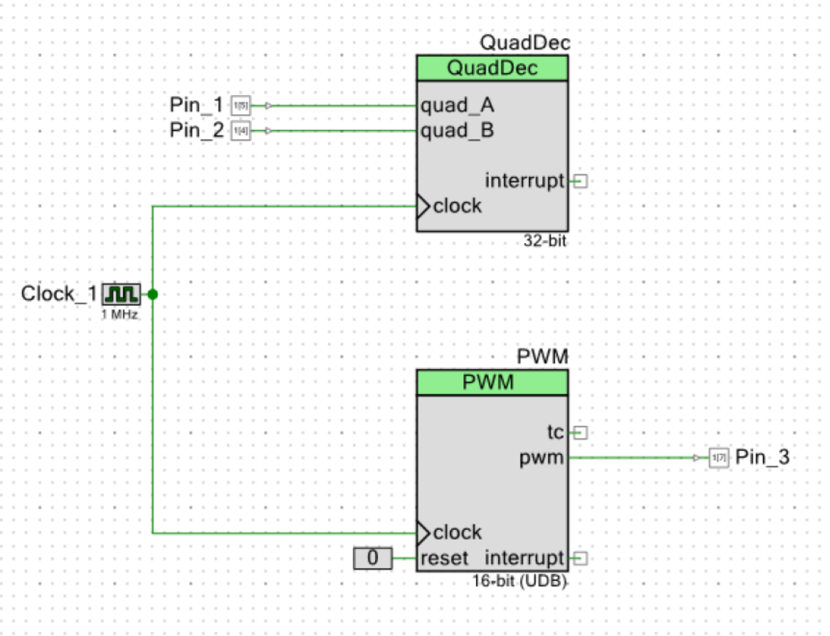
UART



Interrupts



Spinner Motor



State Machine

- 4 States:
  - “S”: Idle
  - “A”: Arm
  - “D”: Disarm
  - “X”: Exit
- ASII character sent to PSoC from LabVIEW before each cycle.

- Only certain routes allowed
  - X -> S
  - A -> X
  - A -> D
  - D -> X
  - S -> X
  - S -> A
- Stepper number and timer period initialized upon receive of new, valid state.

“S” (Idle)

main()

```
RX if receive == 1
    change parameters
    if necessary.
    receive = 0

GetLidarData()

TX if transmit == 1
    Send data to LabVIEW
    transmit = 1
```

Interrupt 1 (20ms)

```
CompareCm
        Object = LOST
Or      Object = FOUND
        Set Direction of Yaw Stepper
Enable Interrupt 2
```

Interrupt 2

```
Step Yaw Stepper in
specified direction
Disable Interrupt 2
```

“A” (Arm)

main()

```
RX if receive == 1
    Calculate step_des &
    rpm_des from last cm value
    Set timer period to 8000
    receive = 0
TX if transmit == 1
    Send data to LabVIEW
    transmit = 0
    receive = 1
```

Interrupt 1 (1ms)

```
Get RPM every 100ms
    Ramp Spinner up to
    rpm_des
```

Interrupt 2

```
Step pitch stepper up to
step_des
Disable Interrupt 2
```

“D” (Disarm)

main()

```
RX if receive == 1
    Set step_des & rpm_des
    to 0.
    receive = 0
TX if transmit == 1
    Send data to LabVIEW
    transmit = 0
    receive = 1
```

Interrupt 1 (1ms)

```
Get RPM every 100ms
    Ramp Spinner down
```

Interrupt 2

```
Step pitch stepper down to 0
When rpm & steps == 0
    Set state to “S”
    Set timer period to 160K
Disable Interrupt 2
```

“X” (Exit)

main()

```
RX if receive == 1
    Set step_des & rpm_des
    to 0.
    receive = 0
TX if transmit == 1
    Send data to LabVIEW
    transmit = 0
    receive = 1
```

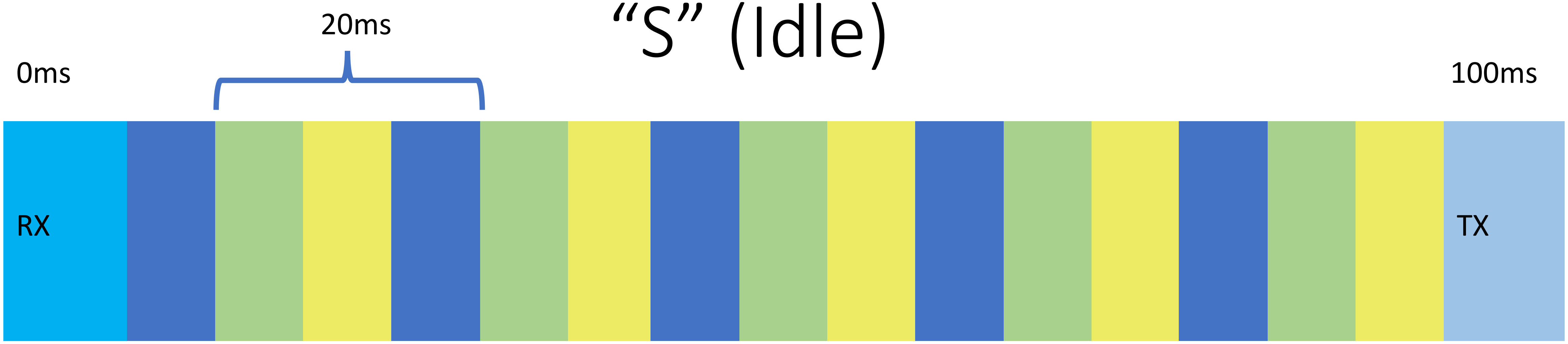
Interrupt 1 (1ms)

```
Get RPM every 100ms
    Ramp Spinner down
```

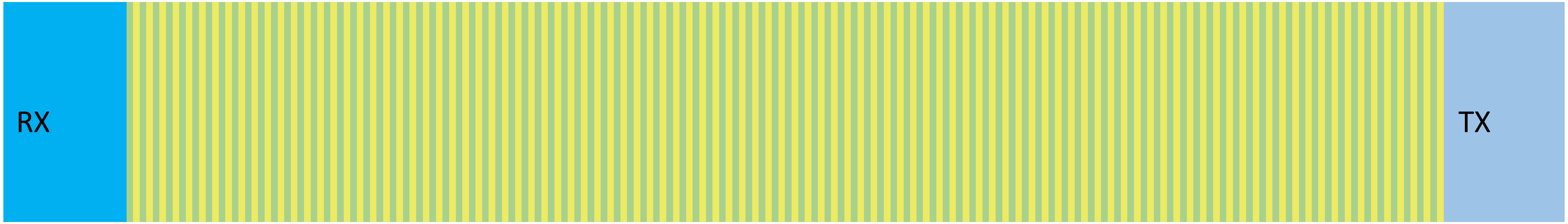
Interrupt 2

```
Step pitch stepper down to 0
Disable Interrupt 2
```

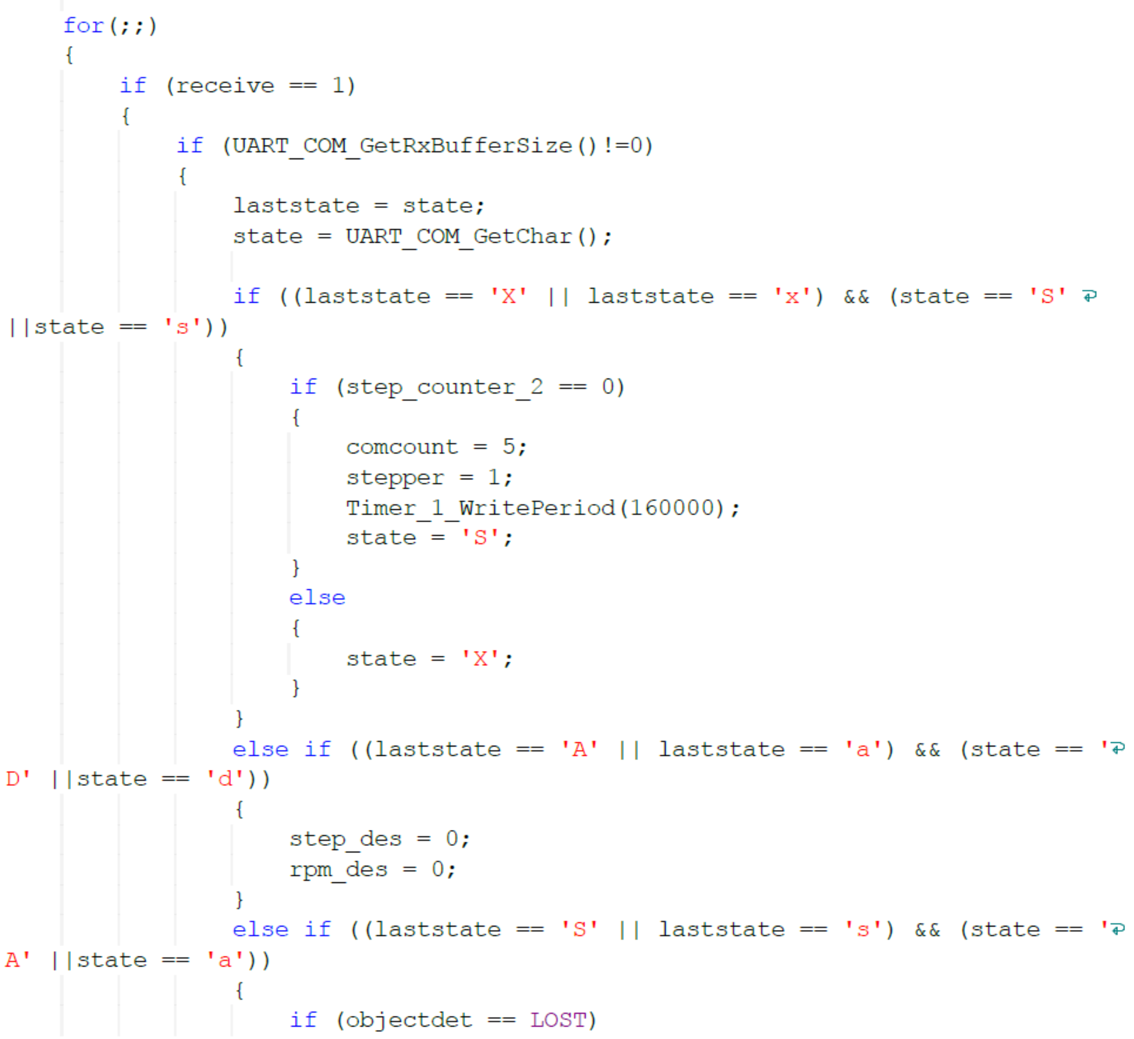
“S” (Idle)



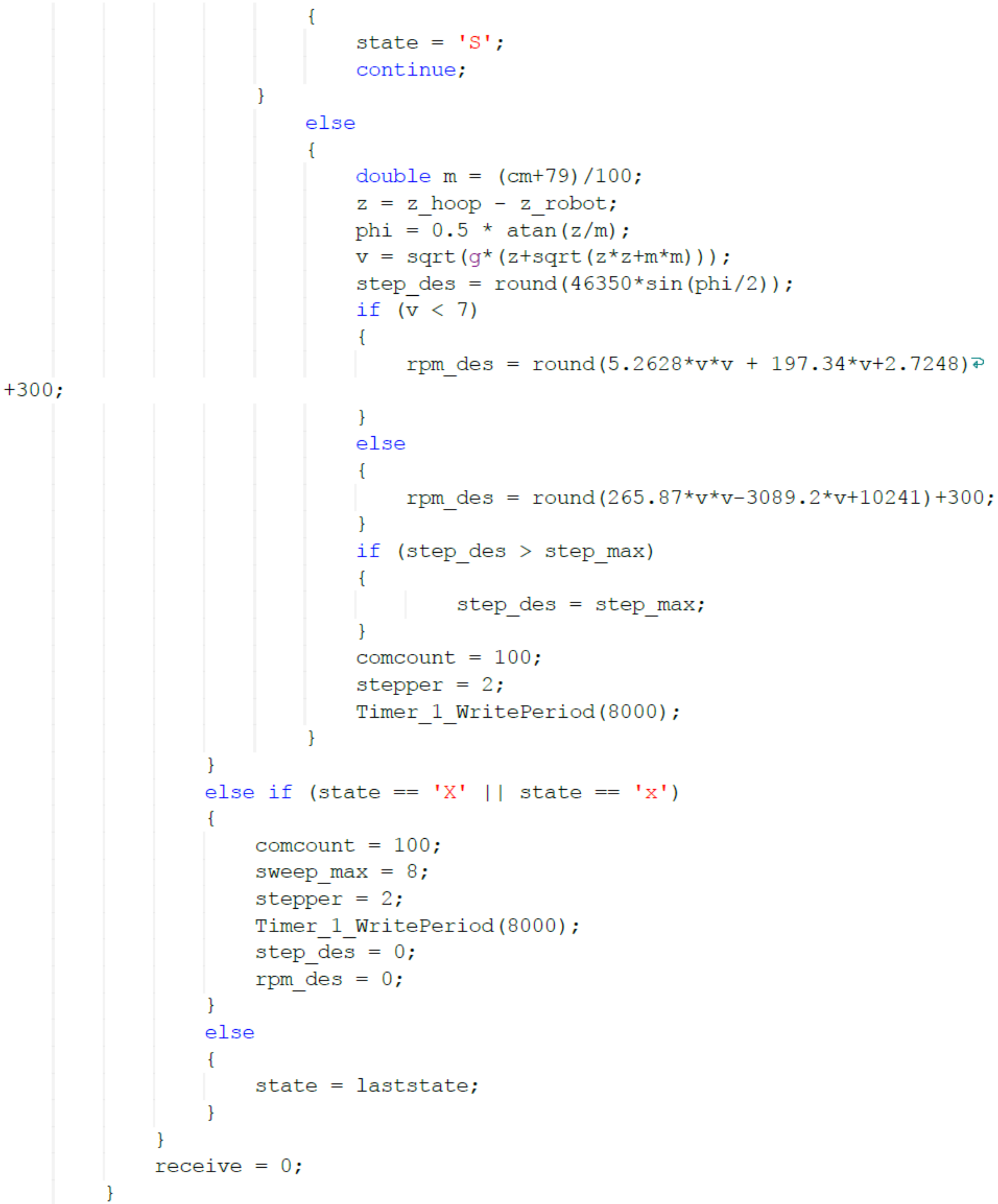
“A” (ARM), “D” (DISARM), “X” (EXIT)



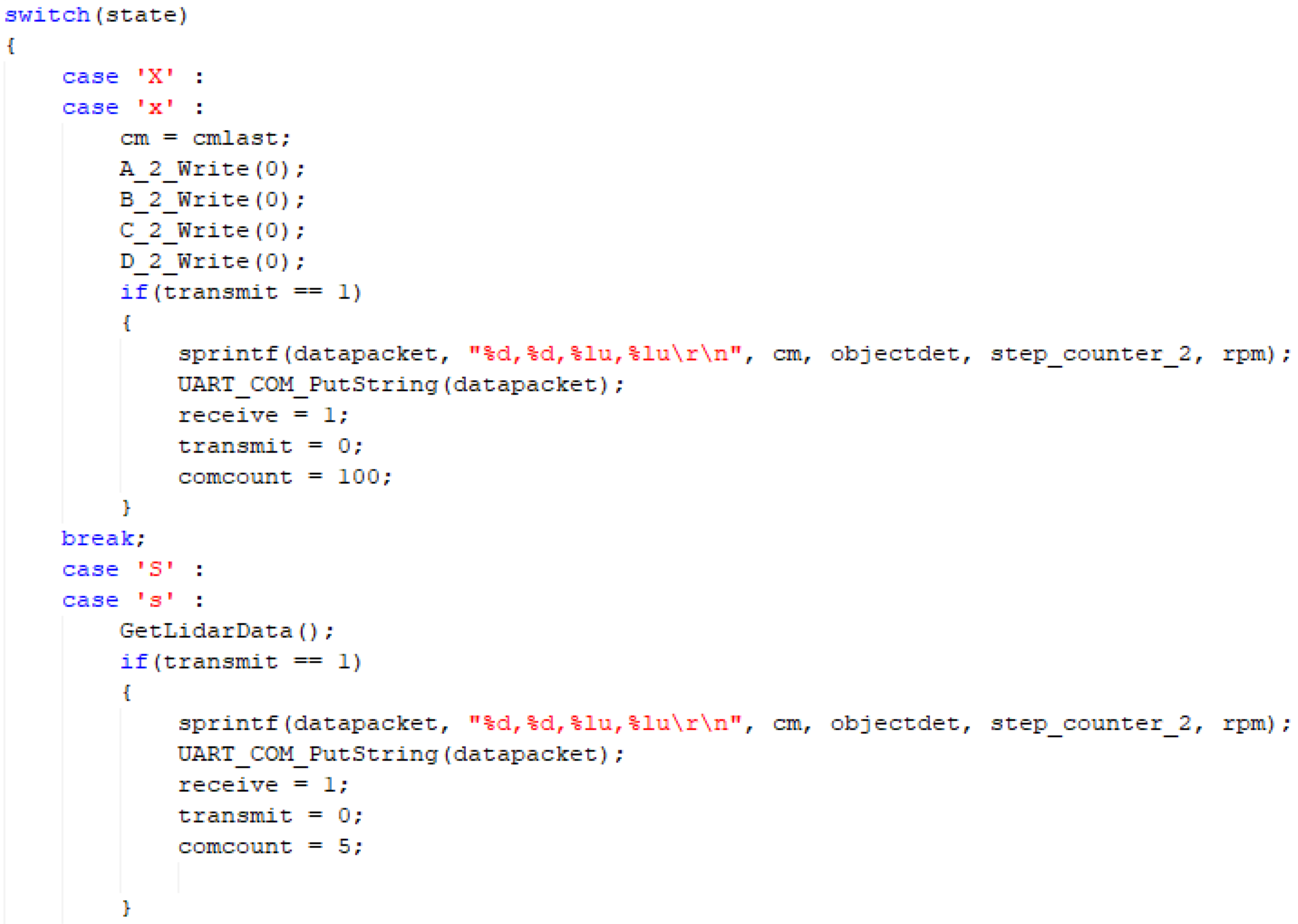
State Transition Code



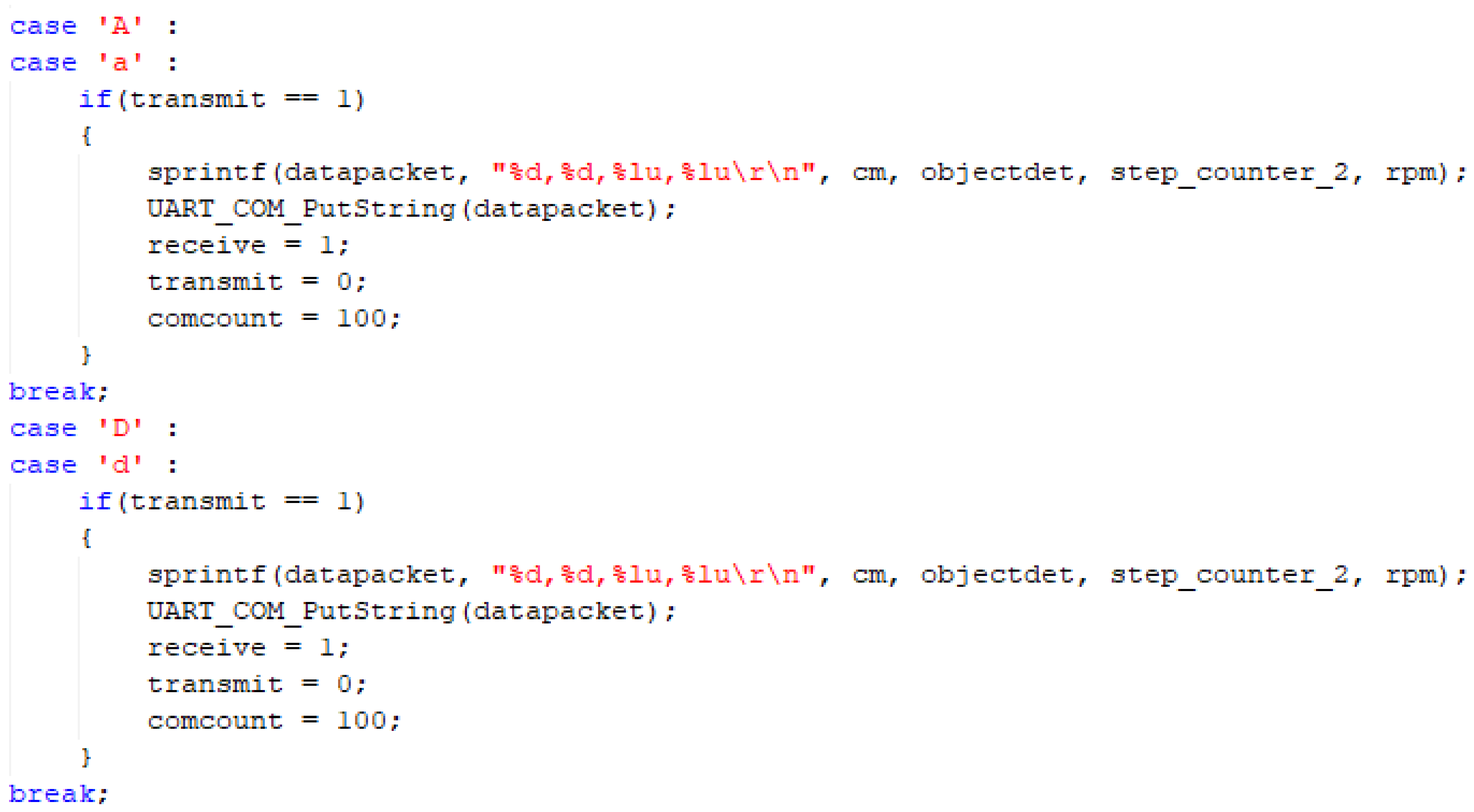
```
    for(;;)
    {
        if (receive == 1)
        {
            if (UART_COM_GetRxBufferSize()!=0)
            {
                laststate = state;
                state = UART_COM_GetChar();

                if ((laststate == 'X' || laststate == 'x') && (state == 'S' ↩
||state == 's'))
                {
                    if (step_counter_2 == 0)
                    {
                        comcount = 5;
                        stepper = 1;
                        Timer_1_WritePeriod(160000);
                        state = 'S';
                    }
                    else
                    {
                        state = 'X';
                    }
                }
                else if ((laststate == 'A' || laststate == 'a') && (state == '↩
D' ||state == 'd'))
                {
                    step_des = 0;
                    rpm_des = 0;
                }
                else if ((laststate == 'S' || laststate == 's') && (state == '↩
A' ||state == 'a'))
                {
                    if (objectdet == LOST)
```



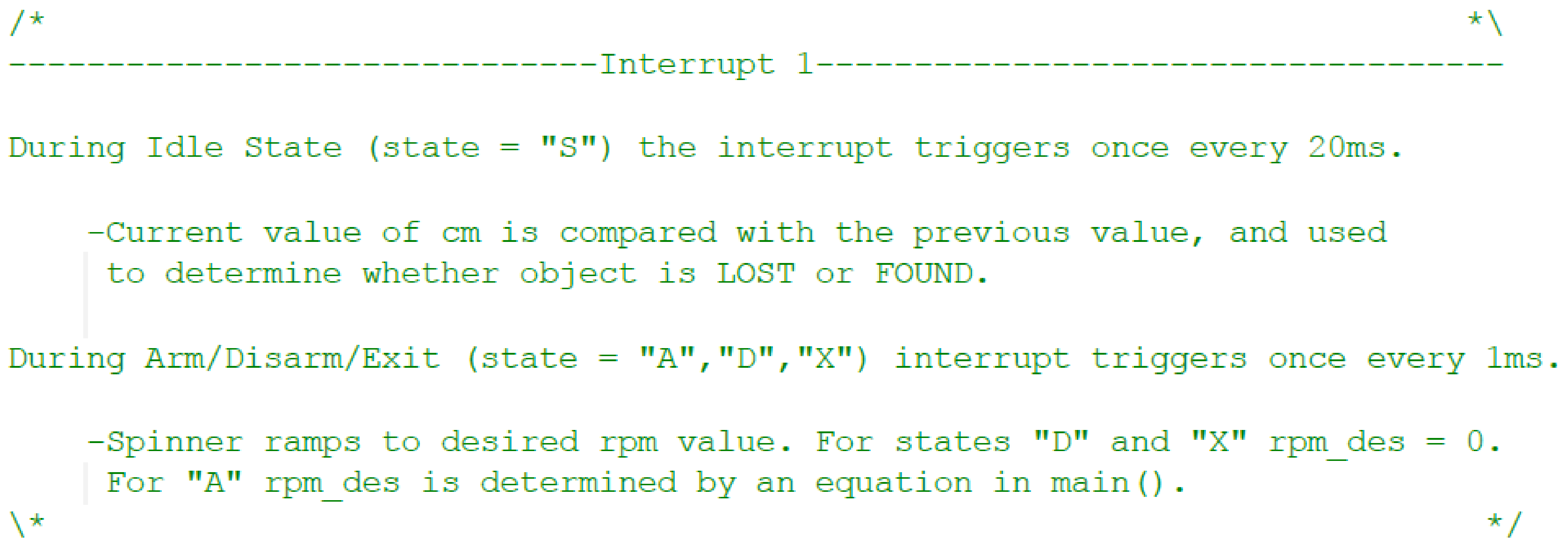
```
                    {
                        state = 'S';
                        continue;
                }
                    else
                    {
                        double m = (cm+79)/100;
                        z = z_hoop - z_robot;
                        phi = 0.5 * atan(z/m);
                        v = sqrt(g*(z+sqrt(z*z+m*m)));
                        step_des = round(46350*sin(phi/2));
                        if (v < 7)
                        {
                            rpm_des = round(5.2628*v*v + 197.34*v+2.7248)↩
+300;
                        }
                        else
                        {
                            rpm_des = round(265.87*v*v-3089.2*v+10241)+300;
                        }
                        if (step_des > step_max)
                        {
                                step_des = step_max;
                        }
                        comcount = 100;
                        stepper = 2;
                        Timer_1_WritePeriod(8000);
                    }
            }
            else if (state == 'X' || state == 'x')
            {
                comcount = 100;
                sweep_max = 8;
                stepper = 2;
                Timer_1_WritePeriod(8000);
                step_des = 0;
                rpm_des = 0;
            }
            else
            {
                state = laststate;
            }
        }
        receive = 0;
    }
```



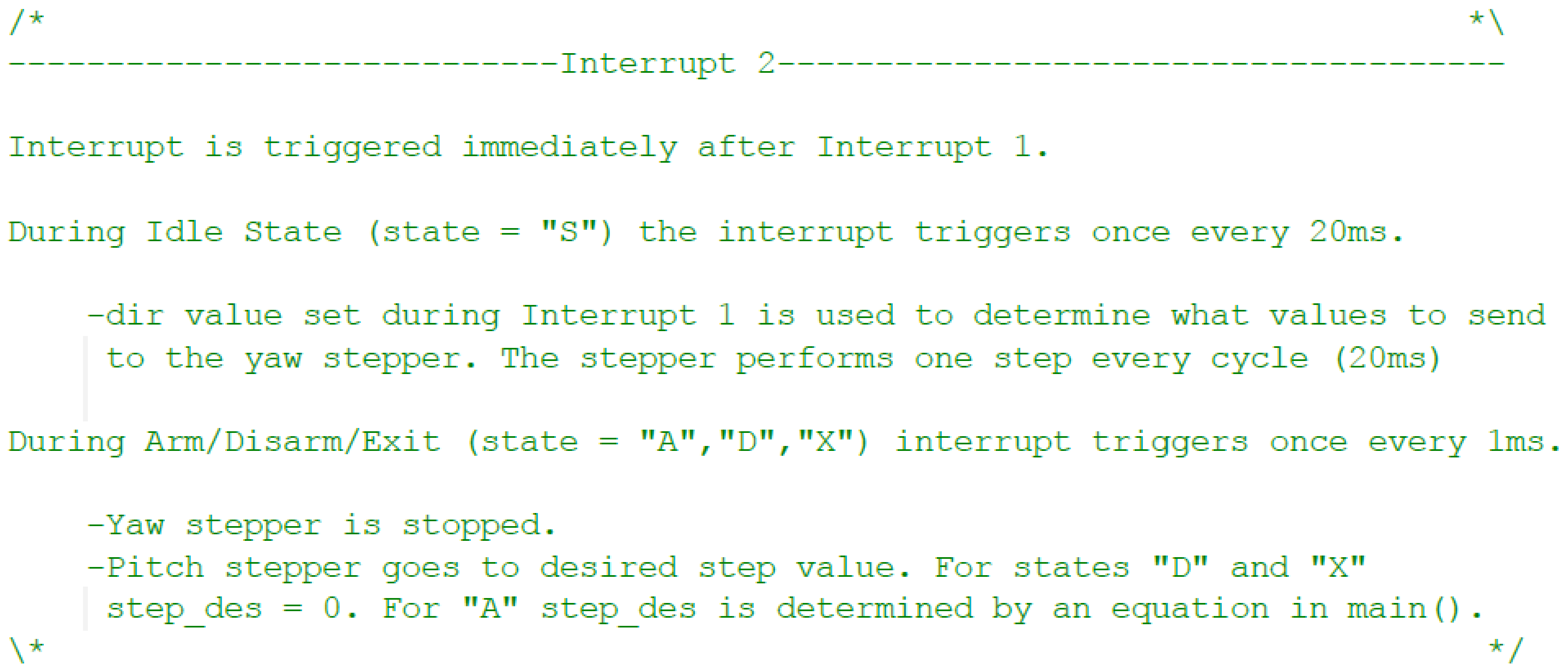
```
switch(state)
{
    case 'X' :
    case 'x' :
        cm = cmlast;
        A_2_Write(0);
        B_2_Write(0);
        C_2_Write(0);
        D_2_Write(0);
        if(transmit == 1)
        {
            sprintf(datapacket, "%d,%d,%lu,%lu\r\n", cm, objectdet, step_counter_2, rpm);
            UART_COM_PutString(datapacket);
            receive = 1;
            transmit = 0;
            comcount = 100;
        }
    break;
    case 'S' :
    case 's' :
        GetLidarData();
        if(transmit == 1)
        {
            sprintf(datapacket, "%d,%d,%lu,%lu\r\n", cm, objectdet, step_counter_2, rpm);
            UART_COM_PutString(datapacket);
            receive = 1;
            transmit = 0;
            comcount = 5;

        }
```



```
case 'A' :
case 'a' :
    if(transmit == 1)
    {
        sprintf(datapacket, "%d,%d,%lu,%lu\r\n", cm, objectdet, step_counter_2, rpm);
        UART_COM_PutString(datapacket);
        receive = 1;
        transmit = 0;
        comcount = 100;
    }
break;
case 'D' :
case 'd' :
    if(transmit == 1)
    {
        sprintf(datapacket, "%d,%d,%lu,%lu\r\n", cm, objectdet, step_counter_2, rpm);
        UART_COM_PutString(datapacket);
        receive = 1;
        transmit = 0;
        comcount = 100;
    }
break;
```



```
/*                                                                    *\
---------------------------Interrupt 1-----------------------------------

During Idle State (state = "S") the interrupt triggers once every 20ms.

    -Current value of cm is compared with the previous value, and used
     to determine whether object is LOST or FOUND.

During Arm/Disarm/Exit (state = "A","D","X") interrupt triggers once every 1ms.

    -Spinner ramps to desired rpm value. For states "D" and "X" rpm_des = 0.
     For "A" rpm_des is determined by an equation in main().
\*                                                                    */
```



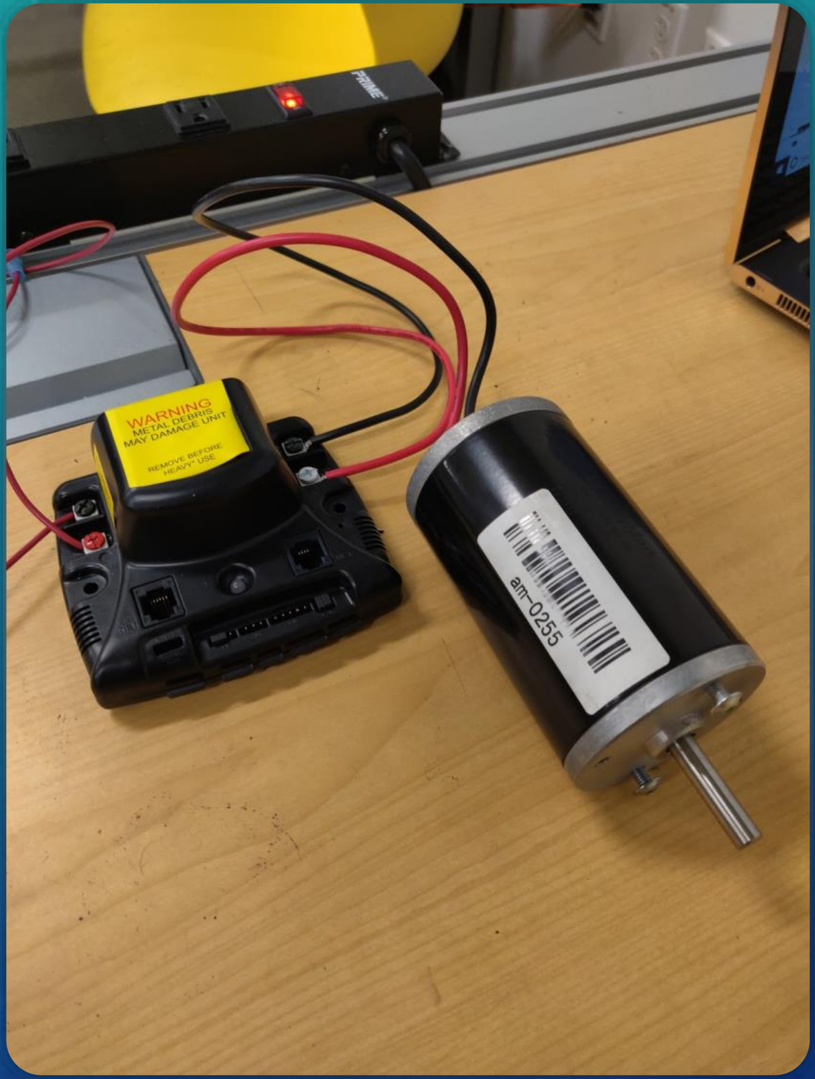
```
/*                                                                          *\
----------------------------Interrupt 2--------------------------------------

Interrupt is triggered immediately after Interrupt 1.

During Idle State (state = "S") the interrupt triggers once every 20ms.

    -dir value set during Interrupt 1 is used to determine what values to send
     to the yaw stepper. The stepper performs one step every cycle (20ms)

During Arm/Disarm/Exit (state = "A","D","X") interrupt triggers once every 1ms.

    -Yaw stepper is stopped.
    -Pitch stepper goes to desired step value. For states "D" and "X"
     step_des = 0. For "A" step_des is determined by an equation in main().
\*                                                                           */
```

Components

- TFMini Micro LIDAR used to determine distance and detect objects.
- Steppers control Yaw and Pitch.
- DC Motor w/ encoder powers spinner.
- 12V battery supplies power to motors.
- Arduino Uno provides 3.3V to the LIDAR.

Spinner Motor and Controller

- Motor: 2.5” DC Brushed CIM motor. Chosen for its high power output and fit within our budget.
- Controller: Jaguar motor controller. Outputs fraction of 12V input from battery to motor via control given PWM from PSoC.



TFMini – Micro LIDAR

- Used to check distance to hoop.
- 0.3m – 12m operating range.
- UART Communication interface

- Byte 1: 0x59
- Byte 2: 0x59
- Byte 3: Low, 8-bit Distance
- Byte 4: High, 8-bit Distance
- Byte 5: Low, 8-bit Strength
- Byte 6: High, 8-bit Strength
- Byte 7: Reserved Byte (nothing)
- Byte 8: Original Signal Quality Degree
- Byte 9: Checksum = Byte 1 + Byte 2 + ...

List of Parts

3D Printed

Laser Cut

Purchased

- Spinner hub
- Right angle holders
- Pitch adjustment holder
- Lead screw attachment

- Frame
- Cross beams
- Center motor holder
- Base
- Base supports
- Gears
- Prototype frame

- Motor x3
- LiDAR
- Sheet metal backing
- Spinner wheel x2
- Metal rod
- Bearings x4
- Lazy Susan
- Hinges
- Casters
- PSOC
- Motor controller x3