Main code gamepad.ino

```
#include <Servo.h>
#include <SoftwareSerial.h>

// Bluetooth module connection
SoftwareSerial BTSerial(0, 1); // RX, TX - note if want to debug, chance these to 2,3 and rewire the bluetooth module so that you can use the serial monitor for debugging also remove// on serial.print and serial.begin commands

// Motor Pin Configuration
const int Rwb = 3;    // Motor controller Pin: In1
const int Rwf = 5;    // Motor controller Pin: In2
const int Lwf = 6;    // Motor controller Pin: In3
const int Lwb = 11;   // Motor controller Pin: In4

// Power variables
int fullPower = 255;

// Servo Configuration
Servo servoOne;
Servo servoTwo;
int posOne = 90;  // Starting position for servo one in degrees
int posTwo = 90;  // Starting position for servo two in degrees

// Catapult-Specific Pins (No overlap with driving system)
const int CATAPULT_SERVO_PIN =10; // Analog pin as digital(10)
//const int MOTOR_PWM = A1; // Launcher motor speed
const int MOTOR_IN1 = A2; // Launcher direction 1
const int MOTOR_IN2 = A3; // Launcher direction 2
const int TRIG_PIN = 8; // Ultrasonic trigger
const int ECHO_PIN = 9; // Ultrasonic echo

// Timing Constants
const int RELOAD_TIME_MS = 1000;
const int SHOOT_TIME_MS = 1000;
const int COOLDOWN_MS = 1000;
const int DETECTION_RANGE_CM = 10;
```

```
Servo catapultServo;

void setup() {
    //Serial.begin(9600); //remove // to debug
    BTSerial.begin(9600);

    // Motor Setup
    pinMode(Rwf, OUTPUT);  //right motor pinmode forward
    pinMode(Rwb, OUTPUT);  //right motor pinmode backward
    pinMode(Lwf, OUTPUT);  //left motor pinmode forward
    pinMode(Lwb, OUTPUT);  //left motor pinmode backward

    // Servo Setup
    servoOne.attach(9);  //Servo one pin
    servoOne.write(posOne);  //Servo one goes to start position
    servoTwo.attach(10); //Servo two pin
    servoTwo.write(posTwo);  //Servo two goes to start position
      // Catapult Servo
  catapultServo.attach(CATAPULT_SERVO_PIN);
  catapultServo.write(0); // Reset position

  // Motor Controller
  pinMode(MOTOR_IN1, OUTPUT);
  pinMode(MOTOR_IN2, OUTPUT);
  //pinMode(MOTOR_PWM, OUTPUT);

  // Ultrasonic Sensor
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  // Initialize motor off
  stopMotor();
}

void loop() {
    if (BTSerial.available()) {
        char command = BTSerial.read(); //reads command from bluetooth app
        //Serial.println(command); //Prints command recieved from
bluetooth app to serial monitor    //remove // to debug
```

```
        switch (command) {
            case 'F': drive("forward", fullPower); break;
            case 'B': drive("backward", fullPower); break;
            case 'L': turn("leftPivot", fullPower); break;
            case 'R': turn("rightPivot", fullPower); break;
            case 'S': servoSwing(&servoOne, 60, 120); break;  // Swing
ServoOne to max position when pressing Square
            case 'C': servoSwing(&servoOne, 60, 120); break;  // Swing
ServoOne to min position when pressing Circle
            case 'T': servoIncremental(&servoTwo, 5, 180); break;  //
Increment ServoTwo up when pressing Triangle
            case 'X': servoIncremental(&servoTwo, -5, 0); break;   //
Increment ServoTwo down when pressing Cross
            default: fullStop(); break;
        }

    }
  int distance = measureDistance();
  if(distance < DETECTION_RANGE_CM && distance > 0) {
    launchSequence();
    delay(COOLDOWN_MS); // Prevent rapid re-triggering
  }
  delay(100);

}
```

Catapult.ino

```
void setup1() {
  // Catapult Servo
  catapultServo.attach(CATAPULT_SERVO_PIN);
  catapultServo.write(0); // Reset position

  // Motor Controller
  pinMode(MOTOR_IN1, OUTPUT);
  pinMode(MOTOR_IN2, OUTPUT);
  //pinMode(MOTOR_PWM, OUTPUT);

  // Ultrasonic Sensor
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
```

```
  // Initialize motor off
  stopMotor();
}

void loop1() {
  int distance = measureDistance();

  if(distance < DETECTION_RANGE_CM && distance > 0) {
    launchSequence();
    delay(COOLDOWN_MS); // Prevent rapid re-triggering
  }
  delay(100);
}

// Distance measurement
int measureDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH);
  return (duration/2) / 29.1; // Convert to cm
}

// Full launch sequence
void launchSequence() {
  reloadCatapult();
  fireCatapult();
}

void reloadCatapult() {
  // Wind motor backward
  digitalWrite(MOTOR_IN1, LOW);
  digitalWrite(MOTOR_IN2, HIGH);
  //analogWrite(MOTOR_PWM, 255);
  delay(RELOAD_TIME_MS);
  stopMotor();
```

```
}

void fireCatapult() {
  // Release servo arm
  catapultServo.write(120);
  delay(100);

  // Unwind motor forward
  digitalWrite(MOTOR_IN1, HIGH);
  digitalWrite(MOTOR_IN2, LOW);
  //analogWrite(MOTOR_PWM, 200);
  delay(SHOOT_TIME_MS);
  stopMotor();

  // Return to ready position
  delay(1000);
  catapultServo.write(0);
}

void stopMotor() {
  digitalWrite(MOTOR_IN1, LOW);
  digitalWrite(MOTOR_IN2, LOW);
  //analogWrite(MOTOR_PWM, 0);
}
```

Driving.ino

```
// Function to move the robot forward/backward
void drive(String dir, int pwr) {
    fullStop();
    if (dir == "forward") {
        analogWrite(Rwf, pwr);
        analogWrite(Lwf, pwr);
    } else if (dir == "backward") {
        analogWrite(Rwb, pwr);
        analogWrite(Lwb, pwr);
    }
}
```

```
// Function to pivot the robot left/right
void turn(String dir, int pwr) {
    fullStop();
    if (dir == "rightPivot") {
        analogWrite(Lwf, pwr * 0.5);
        analogWrite(Rwb, pwr * 0.5);
    } else if (dir == "leftPivot") {
        analogWrite(Rwf, pwr * 0.5);
        analogWrite(Lwb, pwr * 0.5);
    }
}

// Function to stop all motor motion
void fullStop() {
    analogWrite(Rwf, LOW);
    analogWrite(Lwf, LOW);
    analogWrite(Rwb, LOW);
    analogWrite(Lwb, LOW);
}

// Function to swing a servo (go from one position to another without
stopping) - default set to servo 1
void servoSwing(Servo* servo, int minAngle, int maxAngle) {
    static bool direction = true;
    if (direction) {
        servo->write(maxAngle);
    } else {
        servo->write(minAngle);
    }
    direction = !direction;
}

// Function to move a servo incrementally (add or subtract step value to
move servo bit by bit) - default set to servo 2
void servoIncremental(Servo* servo, int step, int limit) {
    int currentPos = servo->read();
    int newPos = constrain(currentPos + step, 0, limit);
    servo->write(newPos);
}
```