Relazione Tecnica progetto
Volterra Urban Preemption System (VUPS)

1. Problema e Soluzione
Il problema: I mezzi di soccorso (ambulanze, vigili del fuoco) perdono tempo prezioso agli incroci regolati da semafori, rischiando collisioni o dovendo attendere il passaggio pedonale.
La soluzione: Un sistema di Preemption (precedenza forzata) che permette al conducente di comunicare via radio con il semaforo. Il sistema mette in sicurezza i pedoni e garantisce la luce verde al veicolo d'emergenza, automatizzando la gestione della precedenza.

2. Descrizione Tecnica e Funzionamento
Il progetto si basa su un'architettura Master-Slave via radiofrequenza:
Modulo TX (Trasmettitore): Utilizza un microcontrollore Arduino collegato a un modulo radio FS1000A a 433MHz. All'invio di un pacchetto dati tramite la libreria RadioHead (modulazione ASK), il sistema trasmette un comando criptato "GO".
Modulo RX (Ricevitore): Un modulo XY-MK-5V riceve il segnale e lo passa a un secondo Arduino che sovrascrive il ciclo semaforico standard.
Logica di commutazione:
Stato di riposo: Il sistema mantiene il verde per i pedoni e il rosso per le auto (strada a senso unico).
Transizione: Ricevuto il segnale, il Verde Pedoni lampeggia per 1 secondo (fase di sgombero) prima di passare al Rosso.
Fase Emergenza: Accertato il rosso pedonale, scatta il Verde Auto per 5 secondi.
Ripristino: Il semaforo auto passa al Giallo (1s) e poi al Rosso. Solo dopo un intervallo di sicurezza, il verde torna ai pedoni.

3. Codice del Progetto
Il codice sfrutta la libreria RH_ASK.h. La logica principale risiede nel ricevitore, che gestisce i tempi tramite la funzione delay() (o millis() per versioni avanzate) per garantire che le sequenze semaforiche non si sovrappongano mai, eliminando il rischio di "verde contemporaneo".

Trasmettitore (tx)

```
#include <RH_ASK.h>
#include <SPI.h>

RH_ASK radio;

const int buttonPin = 2;    // Pulsante attivazione ambulanza
const int ledFeedback = 5;  // LED che conferma l'invio sul telecomando
```

```
void setup() {
   Serial.begin(9600);
   if (!radio.init()) {
      Serial.println("Errore modulo TX");
      while (1);
   }
   pinMode(buttonPin, INPUT_PULLUP);
   pinMode(ledFeedback, OUTPUT);
   Serial.println("Trasmettitore Ambulanza Pronto");
}

void loop() {
   // Se il tasto viene premuto (va a LOW perché INPUT_PULLUP)
   if (digitalRead(buttonPin) == LOW) {
      const char *msg = "GO";
      radio.send((uint8_t *)msg, strlen(msg));
      radio.waitPacketSent();

      digitalWrite(ledFeedback, HIGH); // Accendi led feedback
      Serial.println("Segnale inviato!");

      delay(1000); // Debounce e protezione per invii multipli
      digitalWrite(ledFeedback, LOW);
   }
}
```

Ricevitore (rx)

```
#include <RH_ASK.h>
#include <SPI.h>

RH_ASK radio;

// Pin Semaforo Auto
const int autoRosso = 2;
const int autoGiallo = 3;
const int autoVerde = 4;

// Pin Semaforo Pedoni
const int pedRosso = 5;
const int pedVerde = 6;

void setup() {
   Serial.begin(9600);
   if (!radio.init()) {
      while (1);
   }
```

```
    pinMode(autoRosso, OUTPUT);
    pinMode(autoGiallo, OUTPUT);
    pinMode(autoVerde, OUTPUT);
    pinMode(pedRosso, OUTPUT);
    pinMode(pedVerde, OUTPUT);

    // Stato iniziale: Auto ROSSO, Pedoni VERDE
    statoIniziale();
}

void statoIniziale() {
    digitalWrite(autoRosso, HIGH);
    digitalWrite(autoGiallo, LOW);
    digitalWrite(autoVerde, LOW);
    digitalWrite(pedRosso, LOW);
    digitalWrite(pedVerde, HIGH);
}

void loop() {
    uint8_t buf[RH_ASK_MAX_MESSAGE_LEN];
    uint8_t buflen = sizeof(buf);

    if (radio.recv(buf, &buflen)) {
        // Se riceve il comando "GO"
        if (buf[0] == 'G' && buf[1] == 'O') {
            eseguiSequenzaEmergenza();
        }
    }
}

void eseguiSequenzaEmergenza() {
    // 1. Il verde pedonale lampeggia per 1 secondo
    for(int i=0; i<5; i++) {
        digitalWrite(pedVerde, LOW);
        delay(100);
        digitalWrite(pedVerde, HIGH);
        delay(100);
    }

    // 2. Pedoni diventano ROSSI, Auto diventano VERDI (subito)
    digitalWrite(pedVerde, LOW);
    digitalWrite(pedRosso, HIGH);

    digitalWrite(autoRosso, LOW);
    digitalWrite(autoVerde, HIGH);

    delay(5000); // Rimane verde per le auto per 5 secondi
```

```
    // 3. Auto diventano GIALLE per 1 secondo
    digitalWrite(autoVerde, LOW);
    digitalWrite(autoGiallo, HIGH);
    delay(1000);

    // 4. Auto tornano ROSSE
    digitalWrite(autoGiallo, LOW);
    digitalWrite(autoRosso, HIGH);

    delay(500); // Piccola pausa di sicurezza tra i due rossi

    // 5. Pedoni tornano VERDI
    digitalWrite(pedRosso, LOW);
    digitalWrite(pedVerde, HIGH);
}
```