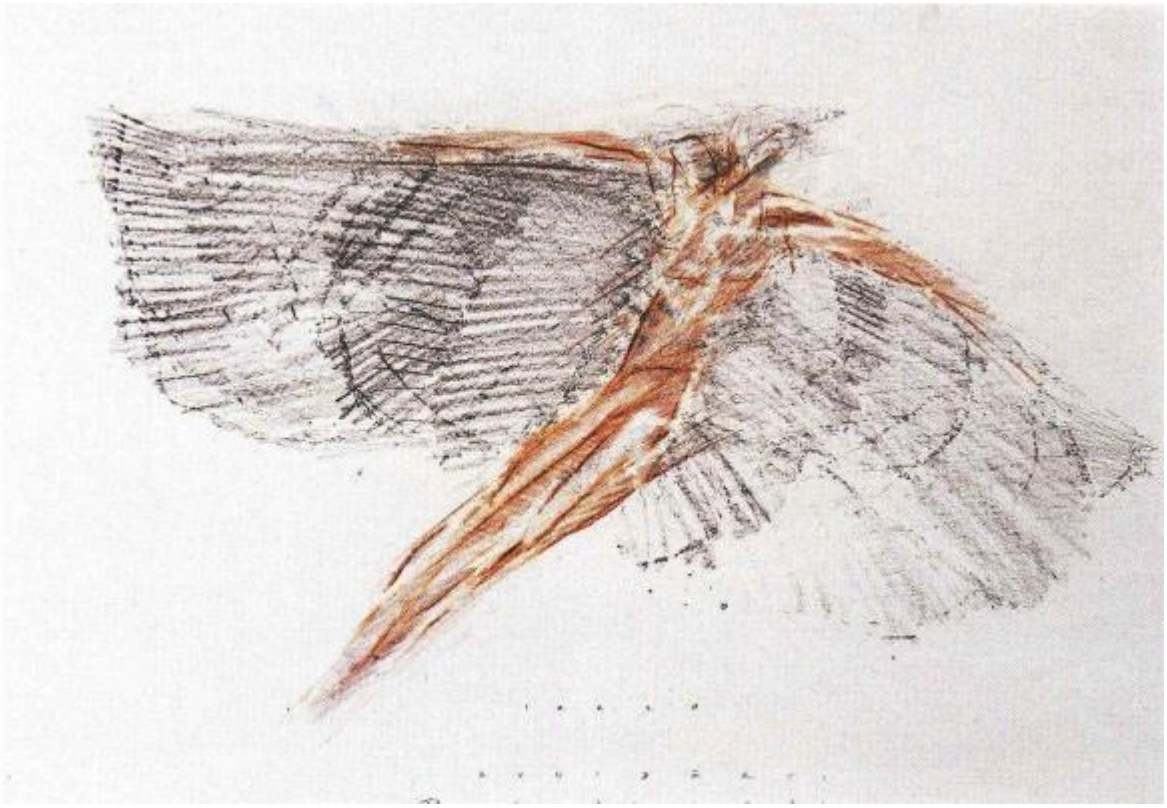
Progetto Vicare



FABRIZIO CACCHIONNI - 2015

INCIPIT

La volontà alla base del progetto è quella di fornire un dispositivo hardware o software automatico di chiamata d'emergenza per motociclisti.
Troppo spesso la mancanza di reattività nel soccorso dopo una caduta, provoca la morte del pilota.
Per esperienza posso affermare che non sempre è possibile per l'incidentato chiamare i primi soccorsi.
Vicare, in base a sensori, rileva se è avvenuta o meno una caduta e invia un SMS ad un numero precedentemente impostato con latitudine, longitudine e ultima velocità rilevata. Può essere installato sul motoveicolo, su uno smartphone, dentro la tuta del pilota.

FUNZIONAMENTO E APPLICAZIONE

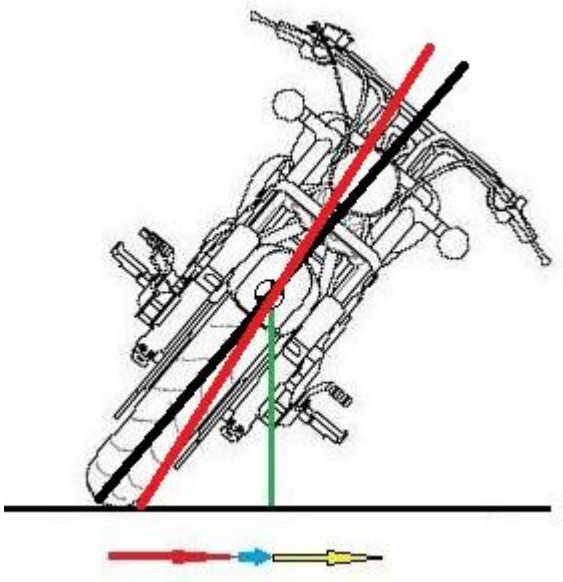
Il tipico esempio è quello dell'installazione sul motoveicolo, ove si è prodotto un primo prototipo funzionante.
In base alla rilevazione di un giroscopio, una scheda verifica in real time i dati di angolo x e y.
Se viene rilevato un grado superiore ai ± 65° per una qualsiasi delle coordinate, la scheda acquisisce la posizione GPS e la invia tramite SMS.
Si identificano quindi le componenti principali di Vicare:

- una scheda di elaborazione dati programmabile;
- un giroscopio multidimensionale;
- un rilevatore GPS;
- un MODEM GSM per l'invio di SMS dotato di scheda SIM.

Un qualsiasi dispositivo contenente questi moduli può dotarsi di Vicare.
Il prototipo funzionante è stato assemblato e programmato dal sottoscritto in totale autonomia. Il test è stato eseguito su motoveicolo Ducati Monster proprietario. Tutti i dati scambiati sono stati ricevuti da dispositivi personali.

Prototipo

Per il prototipo si è usata una scheda di elaborazione Raspberry pi con architettura ARM e sistema GNU/Linux Debian.
Il giroscopio è un SOC a 3 assi MPU-6050, comunicante con bus e moduli free.
Il ricevitore GPS è un Adafruit Ultimate a 66 canali, comunicante via USB e controllato tramite apposito demone GPSD.
Il modem è composto da una Internet Key Huawei, comunicante via usb e moduli free.
L'alimentazione viene fornita direttamente dalla batteria e convertita da 12V DC a 5V DC 2 Amperè tramite adattatore USB.
Il montaggio deve essere particolarmente curato in maniera che l'oscilloscopio risulti essere perfettamente allineato e/o calibrato a 0° per tutti gli assi.
Al collegamento dell'alimentazione, la scheda fa partire il sistema caricato in una memoria micro SD.
Vengono inizializzati tutti i sensori, in particolare viene fatta una finta chiamata al demone GPS per caricare e verificare il collegamento con il ricevitore.
Viene registrata la posizione di partenza dell'oscilloscopio e viene fatto partire il codice in linguaggio Python.
Questo inizierà a controllare in maniera iterativa che gli angoli x e y non superino un dislivello di 65 gradi rispetto alla posizione iniziale.



In tal caso, quando il rollio supera di 65 ° si è caduti, in maniera più o meno accidentale.
Rilevati in tempo reale questi dati, se e solo se si supera l'angolo massimo di rollio, viene acquisto dal ricevitore GPS il punto geografico dove ci si trova.
Viene poi caricata la stringa di testo con tali coordinate e velocità. Questa viene inviata via SMS ad un numero di cellulare prima caricato.
Nel caso invece siamo nella tolleranza, il codice procederà all'analisi degli angoli negli istanti successivi.

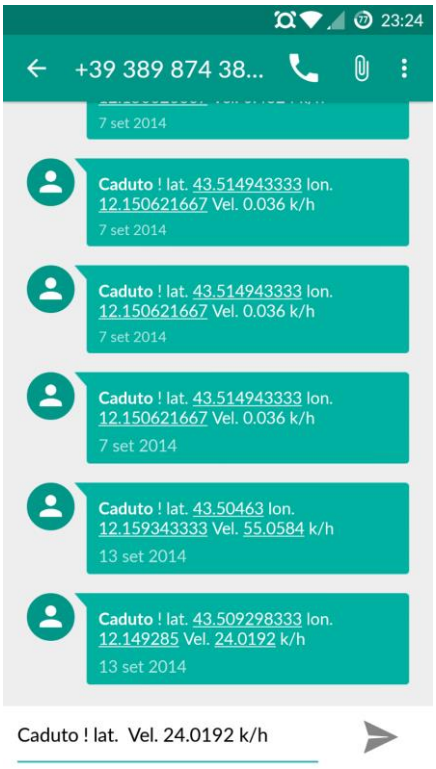
Foto

Codice

```
import time
import math
import mpu6050
import gps
import serial
import datetime

# Inizializzazione sensori
mpu = mpu6050.MPU6050()
mpu.dmpInitialize()
mpu.setDMPEnabled(True)



packetSize = mpu.dmpGetFIFOPacketSize()
# Apertura demone GPS su porta 2947
session = gps.gps("localhost", "2947")
session.stream(gps.WATCH_ENABLE | gps.WATCH_NEWSTYLE)
noncaduto = True
while noncaduto:
    # Lettura INT_STATUS byte
    mpuIntStatus = mpu.getIntStatus()

    if mpuIntStatus >= 2: # Drop pacchetti doppi
        # lettura contatore FIFO corrente
        fifoCount = mpu.getFIFOCount()

        # check overflow del FIFO per evitare blocchi o letture incorrette
        if fifoCount == 1024:
            # reset FIFO
            mpu.resetFIFO()



        # Attesa dati corretti
        fifoCount = mpu.getFIFOCount()
        while fifoCount < packetSize:
            fifoCount = mpu.getFIFOCount()

        result = mpu.getFIFOBytes(packetSize)
        q = mpu.dmpGetQuaternion(result)
        g = mpu.dmpGetGravity(q)
        ypr = mpu.dmpGetYawPitchRoll(q, g)

        #print(ypr['yaw'] * 180 / math.pi),

        angolodxsx=ypr['pitch'] * 180 / math.pi
        #print(angolodxsx)

        angolofxrx=ypr['roll'] * 180 / math.pi
```

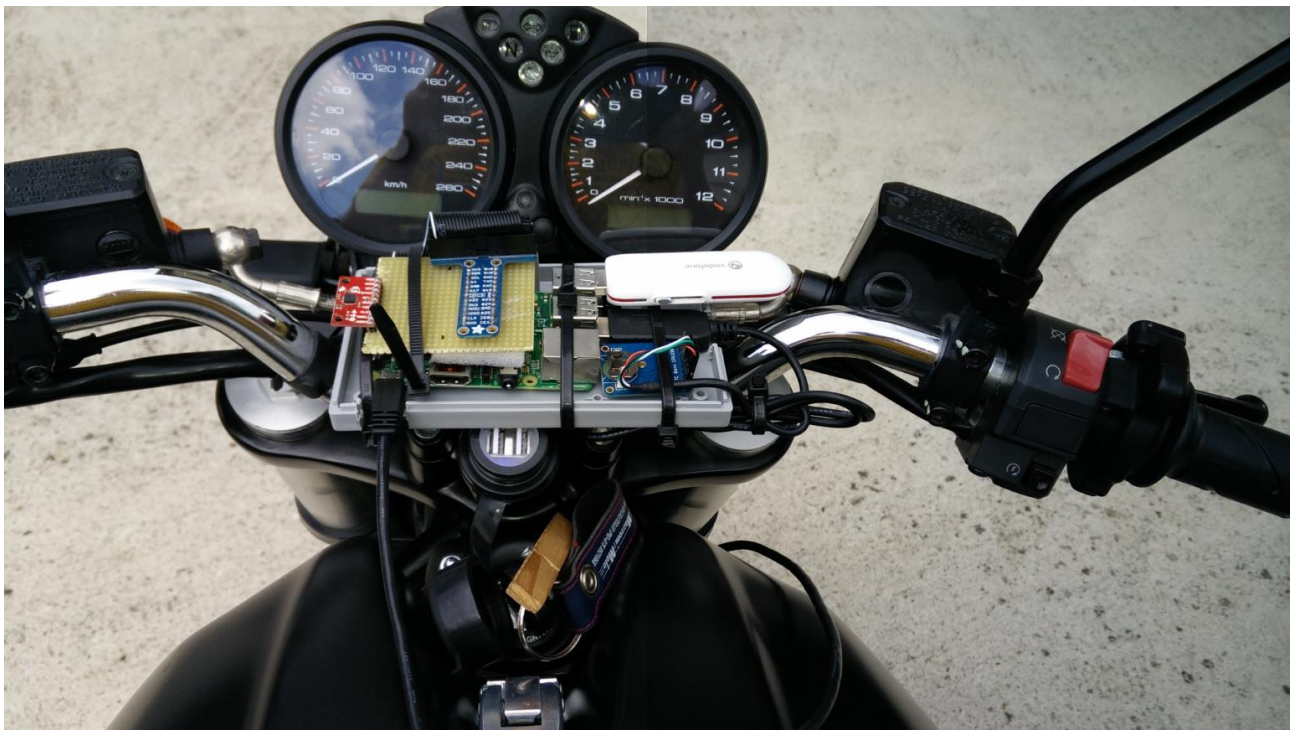
```
        #print(angolofxrx)
        report = session.next()
        if (angolodxsx < -55) or (angolodxsx >55) or (angolofxrx < -55) or (angolofxrx > 55):
             #print("Caduto ! ")
             if report['class'] == 'TPV':
                          if hasattr(report, 'lat'):
                                   latitudine = str(report.lat)
                          if hasattr(report, 'lon'):
                               longitudine = str(report.lon)
                               if hasattr(report, 'speed'):
                                    velocita = str(report.speed * gps.MPS_TO_KPH)
                                    noncaduto = False
                                    recipient = "+39123456789" # numero QUI
                                    message = "Caduto ! lat. " + latitudine + " lon. " +
longitudine + " Vel. " + velocita + " k/h"
                                    print(message)
                                    file=open("/home/pi/gpslog.txt", "a")
                                    file.write("\n" +
str(datetime.datetime.now().strftime("%d/%m/%y %H:%M")) + " - " + message)
                                    phone = serial.Serial("/dev/ttyUSB1",  115200, timeout=5)
                                    try:
                                         time.sleep(0.5)
                                         phone.write(b'ATZ\r')
                                         time.sleep(0.5)
                                         phone.write(b'AT+CMGF=1\r')
                                         time.sleep(0.5)
                                         phone.write(b'AT+CMGS="' + recipient.encode() +
b'"\r')
                                         time.sleep(0.5)
                                         phone.write(message.encode() + b"\r")
                                         time.sleep(0.5)
                                         phone.write(chr(26))
                                         time.sleep(0.5)
                                    finally:
                                         phone.close()
                               #print("Caduto ! lat. " + latitudine + " lon. " + longitudine)


        # verifica termine controlli FIFO
        fifoCount -= packetSize
```

Simulazione risultato in caso di attivazione



Allegati fotografici

Prototipo 1 – componenti agganciati al traliccio e scheda in alto



Prototipo 2 – raccoglimento sul manubrio e alimentazione via USB